Departamento de Ingeniería Telemática
Ingeniería Superior de Telecomunicaciones

Desarrollo de un cliente de vídeo bajo demanda para redes con plano de control SIP/IMS.

PROYECTO FIN DE CARRERA

Autor: Aránzazu García García
Tutor: Iván Vidal Fernández

Leganés, octubre 2013

Título: Desarrollo de un cliente VoD para redes con plano de control SIP/IMS
Autor: Aránzazu García García
Director: Iván Vidal Fernández

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Resumen

Desde hace menos de una década, los avances tecnológicos se han incrementado exponencialmente hasta llegar a día de hoy, en el que la inclusión de las tecnologías en nuestras vidas las ha transformado radicalmente, cambiando uno de los conceptos básicos para el ser humano, que es el concepto de comunicación. Hoy en día, realizar algo tan básico como la visualización de un vídeo sobre una noticia desde *Youtube*, descargarse un vídeo correspondiente a una clase de master online, o tener una televisión inteligente en nuestro hogar que nos permite elegir qué película o serie queremos ver, y cuándo, ha provocado un giro de 180 grados en nuestra sociedad, influyendo desde la manera de estudiar, hasta la manera en la que pasamos nuestro tiempo libre. Según estadísticas recientes (1), el tráfico de vídeo consumido en Internet será el 69% de todo el tráfico consumido en Internet en el año 2017, siendo en la actualidad de un 57% sobre el tráfico total. Según estas cifras, la importancia del contenido de vídeo en Internet queda de manifiesto, convirtiendo a este tipo de tráfico en el predominante de la red.

El objetivo de este Proyecto Fin de Carrera consiste en desarrollar un cliente de vídeo bajo demanda (VoD) para redes con plano de control basado en la arquitectura IMS (*IP Multimedia Subsystem*). Este cliente se ha implementado siguiendo las especificaciones para servicios de Televisión IP (IPTV) desarrolladas por el grupo de trabajo TISPAN de ETSI (*European Telecommunications Standards Institute*). El cliente VoD permitirá a un usuario, utilizando un navegador Web, seleccionar un determinado contenido de vídeo (por ejemplo una película) dentro de un catálogo, y solicitar la transmisión de dicho contenido desde un servidor. Además, el cliente permite controlar la reproducción del contenido de vídeo, pausando o reanudando la misma a petición del usuario. El desarrollo del sistema se ha estructurado en dos planos, uno de control, en el que se utiliza el protocolo SIP (*Session Initiation Protocol*), y un plano de datos, que permite transmitir el contenido del vídeo seleccionado por el usuario. Finalmente, se ha desplegado un entorno de pruebas con una infraestructura IMS real, el cual ha permitido verificar el correcto funcionamiento del desarrollo realizado.

Palabras clave: VoD (*Video on Demand*), IMS (*IP Multimedia Subsystem*), TISPAN (*Telecommunications and Internet converged Services and Protocols for Advanced Networking*), SIP (*Session Initiation Protocol*), RTSP (*Real Time Streaming Protocol*).

Abstract

Since less than a decade ago, the technological progress has exponentially increased until today, when the inclusion of technology in our lives has dramatically transformed them, changing one of the basic human beign concepts, the communication concept. Nowadays, something so basic as accessing to a video on Youtube, downloading a video about an online master class, or having a smart TV in our homes which allows us to choose whichmovie or series do we want to watch, and when, have provoke a 180 degrees change in our society, influence from the way we study, to the way we spend our free time. Taking into account the statistics (1), the consumer Internet video traffic will be 69% of all consumer Internet traffic in 2017, up from 57% percent in 2012. According to this numbers, the importance of the Internet video traffic is greatly high, with future forecasting even better.

The goal of this project is to design and develop a video on demand (VoD) based on networks with a control plane based on the IMS (IP Multimedia Subsystem) architecture. This client is implemented using the specification for IPTV services developed by TISPAN ETSI (*European Telecommunications Standards Institute*). The VoD client allows the user, using a web browser, select a video content from a server through the IMS network. Moreover, this client offers the power to the user of controlling the video by stopping it, or playing it again. The system development has been defined into two planes, a control plane, based on the SIP/SDP (Session Initiation Protocol/Session Description Protocol) protocols to establish the session description, through the IMS network. And a data plane which allows sending the video content selected by the user. Finally, it has been deployed a testing IMS environment, to verify the behavior of the project.

Keywords: VoD (*Video on Demand*), IMS (*IP Multimedia Subsystem*), TISPAN (*Telecommunications and Internet converged Services and Protocols for Advanced Networking*), SIP (*Session Initiation Protocol*), RTSP (*Real Time Streaming Protocol*).

Índice general

Índice de figuras

Índice de tablas

Parte I: Introducción

Capítulo 1

Introducción

1.1 Motivación

Actualmente, el mundo en el que vivimos nos induce a estar rodeados de nuevas tecnologías y a tenerlas integradas, casi de una forma dependiente, en nuestro día a día. Desde el momento en que nos levantamos, hasta el momento en que nos acostamos, nuestro *smartphone*, *tablet* o portátil, nos acompañan cada minuto. Vivimos "*conectados*", lo que significa que, cada vez más, demandamos servicios que estos dispositivos electrónicos pueden proporcionarnos. A este respecto, actualmente están adquiriendo mayor relevancia y popularidad las aplicaciones o servicios multimedia en red. Esto queda de manifiesto en estadísticas recientes, según las cuales, en el año 2012 el tráfico en Internet correspondiente a vídeo representa el 57% del total del tráfico de la red (1). La tendencia se mantiene según las estadísticas (1), que afirman que el tráfico de vídeo consumido en Internet será el 69% de todo el tráfico consumido en Internet en el año 2017. En concreto, el tráfico debido al contenido de vídeo bajo demanda se triplicará con respecto al 2012 en el año 2017, siendo la cantidad de tráfico VoD en 2017 equivalente a seis billones de DVDs por mes (1). La provisión de estos servicios se sostiene por el despliegue de tecnologías de acceso de banda ancha, fija y móvil, que proporcionan al usuario acceso a Internet en cualquier momento y desde cualquier localización.

A consecuencia de este resultado, surgen una amplia variedad de desarrollos para establecer arquitecturas de provisión de servicios de vídeo en las redes actuales y en las redes futuras. A este respecto, una iniciativa especialmente relevante es el trabajo de estandarización realizado por el grupo TISPAN (Telecommunications and Internet converged Services and Protocols for Advanced Networking) de ETSI (*European Telecommunications Standards Institute*). Entre otras actividades, TISPAN actualmente trabaja en la definición de un servicio de televisión IP (IPTV) para redes basadas en IMS(*IP Multimedia Subsystem*). IMS es una de arquitectura de señalización desarrollada

por el 3GPP (*3rd Generation Partnership Project*) para dar soporte a la provisión de servicios multimedia en las redes de próxima generación.

Bajo este contexto, el objetivo de este Proyecto Fin de Carrera consiste en el desarrollo de un cliente de vídeo bajo demanda (VoD) para redes con plano de control basado en la arquitectura IMS. Este cliente se ha implementado siguiendo las especificaciones para servicios IPTV desarrolladas por TISPAN (2) y (3). Este cliente VoD permitirá a un usuario, utilizando un navegador Web, seleccionar un determinado contenido de vídeo dentro de un catálogo, y solicitar la transmisión de dicho contenido desde un servidor a través de una red IMS. Además, el cliente permitirá controlar la reproducción del contenido de vídeo, pausando o reanudando la misma a petición del usuario.

1.2 Objetivos

El objetivo principal de este proyecto es el desarrollo de un cliente de vídeo bajo demanda para redes con plano de control IMS, basado en las especificaciones de TISPAN 182 027 (2) y 183 063 (3) para sistemas IPTV. Profundizando este objetivo, podemos destacar los siguientes puntos a tener en cuenta en el desarrollo del sistema:

- El cliente VoD será accesible al usuario mediante un navegador Web. De este modo, se facilita el acceso del usuario al servicio, puesto que no es necesario instalar software específico para la ejecución del mismo, estando disponible desde cualquier equipo que disponga de un navegador.

- El cliente VoD permitirá al usuario seleccionar un contenido de vídeo, y reproducir dicho contenido.

- El desarrollo se estructurará en dos planos: un plano de señalización o control, en el que se utilizará el protocolo SIP, y un plano de datos, que permitirá la transmisión del contenido de vídeo seleccionado por el usuario desde un servidor mediante el protocolo RTP.

- El cliente VoD permitirá controlar la reproducción del contenido de vídeo, pausando o reanudando la misma a petición del usuario.

- El desarrollo realizado será compatible con los procedimientos de registro de usuario y establecimiento de sesión de IMS, según han sido definidos por el 3GPP.

- Para el desarrollo de la aplicación se utilizarán herramientas de tipo *opensource*.

- El cliente VoD será desarrollado para que sea lo más flexible y adaptable posible, ante una futura evolución de la misma.

1.3 Estructura de la memoria

Este documento se encuentra dividido en varios capítulos, los cuales, pasan a describirse a continuación:

- Capítulo 1. Introducción: en este capítulo se describe el marco en el que se establece el proyecto. Se explican los motivos por los cuales se desarrolla, así como los objetivos que se quieren conseguir.

- Capítulo 2. Estado del Arte: describe el conjunto de tecnologías usadas para el desarrollo del proyecto.

- Capítulo 3. Diseño del sistema VoD: en este apartado se describe el diseño del cliente VoD que se ha desarrollado, así como su integración en la arquitectura IPTV definida por TISPAN. También se detallan los distintos procedimientos implementados por el cliente VoD, tanto en el plano de control como en el de datos.

- Capítulo 4. Implementación de los elementos del sistema: en este apartado se describe la implementación del cliente VoD, teniendo en cuenta los requisitos definidos en el apartado anterior, y utilizando las tecnologías descritas en el estado del arte.

- Capítulo 5. Escenario de pruebas: define el escenario de pruebas que se ha utilizado para la validación del desarrollo realizado. Se describe tanto la ejecución del cliente VoD como los resultados obtenidos.
-
- Capítulo 6. Conclusiones y líneas futuras: se describen en este apartado las conclusiones obtenidas tras el desarrollo del cliente VoD, así como las líneas de trabajo futuras de que se pueden seguir a partir del mismo.

- Anexos: se facilita en este apartado, una ampliación de información acerca del sistema desarrollado como extensión a lo tratado en el propio documento. Consta de los siguientes apartados:

    - Anexo A. Presupuesto
    - Anexo B. Manual de usuario
    - Anexo C. Configuración de VLC como servidor de vídeo
    - Anexo D. Glosario

- Bibliografía: incluye el listado de todas las referencias, documentos y libros que han sido utilizados como material de soporte y apoyo para el desarrollo del proyecto.

Parte II: Estado del Arte

Capítulo 2

Estado del Arte

2.1 *SIP*

2.1.1 Introducción:

El protocolo SIP (*Session Initiation Protoco*l, o protocolo de inicio de sesión), (4), es un protocolo de señalización y control, desarrollado por el IETF (5), para la creación, modificación y terminación de sesiones entre dos puntos finales de comunicación. Adicionalmente, SIP permite el establecimiento de sesiones en las que están involucradas más de dos partes (sesiones multiusuario), utilizando elementos intermedios definidos en el protocolo. Este tipo de sesiones, son, por lo general, de tipo multimedia, como por ejemplo, sesiones de vídeo o mensajería, y sesiones de voz sobre IP (*Voice over IP*, *VoIP*). Las principales funciones de SIP son las siguientes:

- Localización de Usuario: determinación del sistema, o sistemas finales utilizados en la comunicación.
- Disponibilidad de Usuario: determinación de la capacidad de la parte llamada de participar en la comunicación.
- Intercambio de capacidades: negociación del medio y de los parámetros de la comunicación.
- Gestión de la sesión: modificación y terminación de sesiones multimedia existentes.

A pesar de todo, SIP no está integrado verticalmente en un sistema de comunicaciones, es decir, no proporciona servicios, sino que ofrece primitivas que pueden ser usadas para implementar dichos servicios, por eso necesita ser combinado con otros protocolos con el objetivo de proveer de un servicio completo a los usuarios.

Algunos de estos protocolos son RTP (*Real Time Protocol*) para proporcionar a los usuarios el transporte de datos con características en tiempo real, por ejemplo audio y vídeo; utilización de RTSP (*Real Time Streaming Protocol*) para controlar servicios basados en *streaming*, como por ejemplo, servicios de vídeo bajo demanda; o SDP (*Session Description Protocol*) para la descripción de sesiones multimedia.

La sintaxis de sus operaciones es muy similar a la establecida por otros protocolos como HTTP (*Hypertext Transfer Protocol*) o SMTP (*Simple Mail Transfer Protocol*), debido a que SIP fue inicialmente diseñado para que la telefonía fuera un servicio más basado en Internet. A finales del año 2000, SIP fue establecido como el protocolo de señalización del 3GPP y de las redes IMS, en detrimento de H.323 o IAX2. De esta forma, hoy en día, es el protocolo adoptado por el 3GPP y el ETSI como protocolo de señalización en las redes 3G y 4G.

2.1.2 Identificación de usuarios:

SIP identifica a los usuarios mediante las denominadas URIs SIP (*Uniform Resource Identifier*) que tienen el formato que se muestra a continuación, según se describe en (4):

sip: [userinfo] hostport [parameters]

- userinfo: es la información del usuario, seguido del carácter "@".
- hostport: indica un nombre de dominio o dirección IP. Adicionalmente, se puede incluir un número de puerto, separándolo del dominio mediante el carácter ":".
- parameters: indica parámetros adicionales. Se indican tras el carácter ";".

Un ejemplo de URI SIP se muestra a continuación:

sip:alice@163.117.140.20:5060

2.1.3 Agentes de Usuario SIP

El estándar de SIP define varios componentes en su arquitectura así como varias formas de implementarlos para el establecimiento de la comunicación. Estos componentes se pueden dividir en dos grupos:

- Agentes de usuario (UAC/UAS)
- Servidores SIP (Proxy, Registrar, Redirect y B2BUA)

A continuación, se hace una breve descripción de cada uno de ellos.

2.1.3.1 Agentes de Usuario (UA, User Agents)

Son utilizados por los usuarios o aplicaciones como puntos finales del protocolo, es decir, son los que producen y consumen los mensajes SIP (ej. teléfonos fijos y móviles, *softphones*, ordenadores, televisores, etc.).

Los agentes de usuario pueden comportarse como clientes (UAC: *User Agent Clients*) o como servidores (UAS: *User Agent Servers*). Un UAC inicia solicitudes de SIP y procesa las respuestas a dicha solicitud. Por su parte, el UAS recibe solicitudes de SIP y genera las respuestas correspondientes. En la Ilustración 1 se muestra un ejemplo de comunicación SIP entre dos Agentes de Usuario, correspondientes a los usuarios Alice y Bob. En este caso, el UA de Alice actúa como UAC, mientras que el UA de Bob actúa como UAS. La comunicación SIP tiene lugar a través de un elemento intermedio (este tipo de elementos serán tratados a continuación).

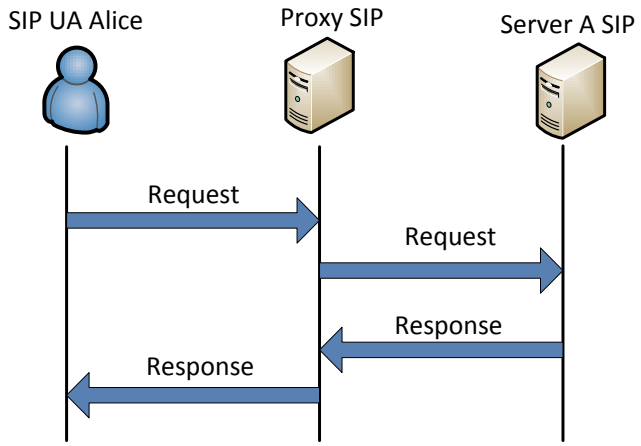

Ilustración 1 Agentes de Usuario SIP

2.1.3.2 Servidores SIP

En este apartado se describen los distintos tipos de servidores definidos en la especificación de SIP (4).

2.1.3.2.1 Servidores Proxy

Estos elementos se utilizan para actuar como nodos intermedios que encaminan los mensajes entre un agente de usuario cliente y un agente de usuario servidor. Los servidores tipo proxy pueden tomar decisiones en cuanto al encaminamiento de las solicitudes SIP, reenviar dichas solicitudes a más de un usuario o servidor final, e incluso, pueden llegar a modificarla antes de enviarla. En general, estos servidores tienen acceso a un servicio de localización que les indica el siguiente salto de la solicitud. En la Ilustración 1 se muestra un ejemplo de servidor Proxy, que encamina las solicitudes y respuestas intercambiadas entre los Agentes de Usuario de Alice y Bob.

Es posible también que un servidor proxy envíe una solicitud a más de un usuario final (ver ejemplo en Ilustración 2). En este caso, se dice que el servidor proxy está actuando como un *forking proxy*.

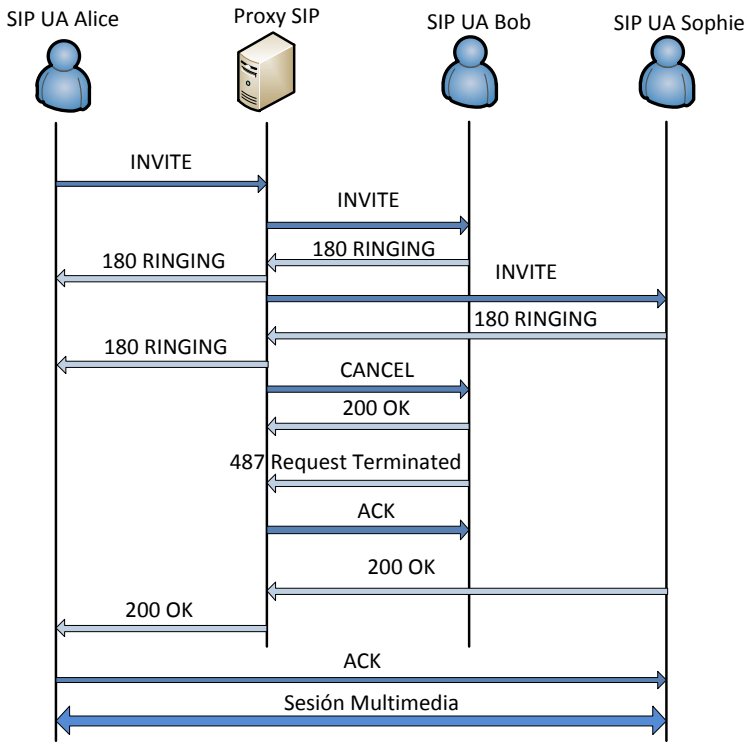

Ilustración 2 Servidor SIP forking proxy

Los servidores proxy pueden ser de dos tipos:

- *Stateless* (sin memoria): procesa las solicitudes SIP según el contenido del mensaje de la solicitud. Una vez ha reenviado el mensaje, el servidor no guarda ningún tipo de información referente al mismo.

- *Stateful* (con memoria): mantiene información referente a las solicitudes y las respuestas que recibe. De esta forma, cuando envía una solicitud inicia un *timer*, y si no le llega respuesta a dicha solicitud, vuelve a reenviarla.

2.1.3.2.2 Servidores Redirect

Los servidores SIP redirectores, actúan como nodo intermedio que informa de la localización de un servidor o usuario. Un mismo servidor puede actuar como Proxy o como Redirect dependiendo del contexto de la comunicación.

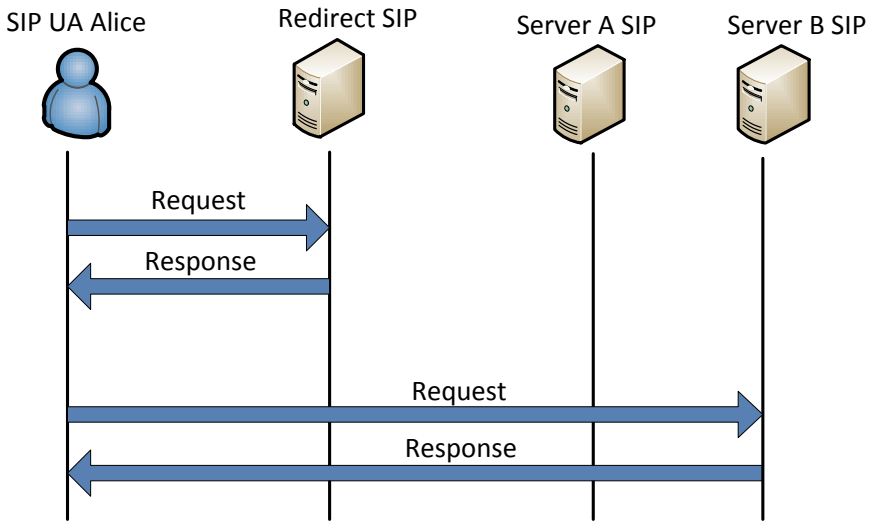

Ilustración 3 Servidor SIP Redirect

En la Ilustración 3 se muestra un ejemplo de operación de un servidor Redirect. En este caso, el servidor recibe una solicitud desde el Agente de Usuario de Alice, y responde la solicitud indicando la localización correspondiente al destino de la misma (esta respuesta incluye la dirección IP y puerto en el que el destino atiende solicitudes). De este modo, el Agente de Usuario iniciador puede reintentar la solicitud directamente hacia el destino.

2.1.3.2.3 Servidores SIP de registro o Registrars

Los servidores de registro tienen como funcionalidad registrar de la localización de un usuario. Para ello se valen de que en el protocolo SIP cada usuario tiene asociada una dirección lógica (esto es, una URI SIP) de la forma *sip:usuario@dominio* (*URI SIP*), que el usuario hace pública a sus contactos y que permite establecer con él una sesión multimedia independientemente de su localización.

Por otro lado, cada vez que el usuario se conecta desde una dirección IP, el agente de usuario que reside en ese terminal envía una solicitud específica de SIP (una solicitud de tipo REGISTER) a un servidor Registrar informando a qué dirección IP (y puerto de nivel de transporte) debe asociarse la URI SIP pública del usuario. Esta asociación se almacena en un servicio de localización y se hace disponible a los *proxies* del dominio, de modo que pueden hacer llegar solicitudes de SIP a la localización real del usuario. En la Ilustración 4 se indica un ejemplo de registro.

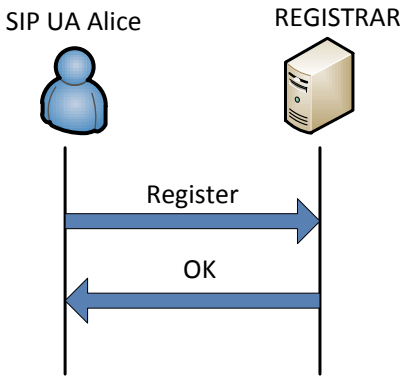

Ilustración 4 Servidor SIP de registro

2.1.3.2.4 Servidores B2BUA

Los servidores Back To Back User Agent reciben peticiones de usuario que reformulan y reenvían como una nueva petición a otros usuarios. Las respuestas son de nuevo recibidas por el B2BUA, quien vuelve a reformularlas y las envía al usuario inicial que realizó la petición. La Ilustración 5 muestra un ejemplo:

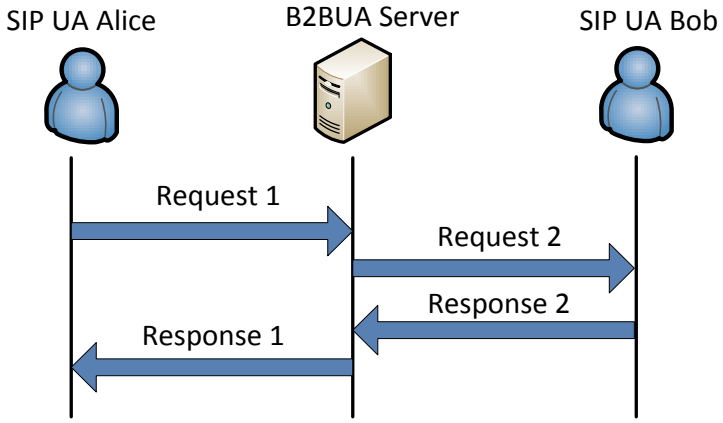

Ilustración 5 Servidor SIP B2BUA

En varios aspectos, el comportamiento de un B2BUA es similar a un proxy. Sin embargo el B2BUA tiene mayor libertad de operación, pudiendo desconectar un lado de la sesión, reconectarlo con otro, iniciar sesiones independientemente o terminarlas sin intervención de los usuarios, modificar la descripción de una sesión multimedia, etc.

2.1.4 Mensajes SIP

Los mensajes de SIP pueden ser de uno de dos tipos, solicitudes o respuestas. Todos los mensajes de SIP se codifican en texto (es decir, son "legibles"), de manera similar a otros protocolos bien conocidos como HTTP (*Hypertext Transfer Protocol*). Cada mensaje consta de tres partes: una línea inicial (que permite distinguir si el mensaje es una solicitud o una respuesta), un conjunto de campos de cabecera y un cuerpo de mensaje opcional.

2.1.4.1 Solicitudes SIP

La primera línea de un mensaje de solicitud se denomina *Request-line*, y su formato, definido en (4), se muestra a continuación:

<Method> <Request URI> <SIP version>

- *Method*: indica el tipo de solicitud.
- *Request URI*: es una URI SIP que indica el destinatario de la petición.
- *SIP version*: indica la versión de SIP.

Las solicitudes principales, definidas inicialmente en SIP (4) se enumeran a continuación:

- *INVITE:* solicita el establecimiento de una sesión multimedia desde un Agente de Usuario.

- *REGISTER:* permite asociar la URI SIP pública de un usuario con su información de localización, mediante un servidor Registrar.

- *BYE:* finaliza una sesión multimedia establecida.

- ACK: permite confirmar el establecimiento de una sesión multimedia.

- CANCEL: utilizado para cancelar solicitudes INVITE de establecimiento de sesión pendientes (esto es, que no han sido respondidas de forma definitiva por el destinatario del INVITE).

- OPTIONS: solicita información sobre las capacidades de un UA o servidor.

Además de estas solicitudes, existen otras definidas en extensiones a la especificación de SIP, tales como: NOTIFY, UPDATE, PRACK, REFER, SUBSCRIBE, MESSAGE e INFO. Estas solicitudes sin embargo, no se describen en el presente estado del arte.

2.1.4.2 Respuestas SIP

En SIP, las respuestas son generadas por un UAS, como contestación a una petición realizada por un UAC. Las posibles respuestas a las peticiones explicadas anteriormente pueden ser (4):

Código	Tipo	Descripción
100-199	Mensaje provisional o de información.	La petición ha sido recibida y está siendo procesada.
200-299	Respuesta de éxito.	Recepción de petición exitosa y aceptada.
300-399	Respuesta de redirección.	Otro tipo de acciones necesitan ser realizadas para seguir adelante con la

		aceptación de la petición.
400-499	Respuesta de fallo de método, error en el cliente.	La petición contiene una sintaxis malformada o no puede ser tratada por el servidor al que llega.
500-599	Respuesta de error en el servidor.	El servidor genera un fallo al intentar tratar una petición aparentemente correcta.
600-699	Respuesta de fallo global.	La petición no puede ser procesada por ningún servidor.

Tabla 1 Respuestas SIP

2.1.4.3 Campos de cabecera de SIP

Cada mensaje de SIP (solicitud o respuesta) incluye un conjunto de campos de cabecera. A continuación se describen algunos de los campos de cabecera de SIP más relevantes:

- Via: mantiene un registro de los elementos atravesados por una solicitud (ej. proxies), y se utiliza para encaminar las respuestas correspondientes a dicha solicitud. A modo de ejemplo:

  Via:SIP/2.0/UDP dominio.com:5060;branch=j5dT6js72akbuf6

- *From*:URI SIP del iniciador de una solicitud, por ejemplo:

  From: Alice<sip:alice@dominioAlice.com>;tag=1922f

- *To*: especifica el receptor lógico de la solicitud, por ejemplo:

  To: Bob <sip:bob@dominio.com>;tag=28619

- *Call-Id*: es un identificador único para cada solicitud INVITE, por ejemplo:

  Call-ID: 5541f955-8dec-11x9-s779-00b0f92r4gj8@dominio.com

- *Cseq*: contiene un número de secuencia y un nombre de método. Permite ordenar solicitudes y diferenciar entre nuevas solicitudes y retransmisiones. A modo de ejemplo:

  Cseq: 1 INVITE

- *Max-Forwards*: utilizado para la detección de bucles en el encaminamiento de las solicitudes. Es iniciado a un valor que va siendo decrementado por cada proxy que atraviesa. Si este valor llega a cero, la solicitud es descartada.

Max Forwards: 10

- *Contact*: indica la dirección de contacto del usuario o recurso generador del mensaje

  Contact: sip:bob@163.117.139.210

- *Expires*: indica el tiempo durante el cual la solicitud y su mensaje son válidos.

  Expires: 30

2.1.4.4 Ejemplo de operación

A continuación se muestra un ejemplo básico de establecimiento de sesión multimedia basado en SIP entre dos usuarios a través de un servidor Proxy, obtenido de (6):

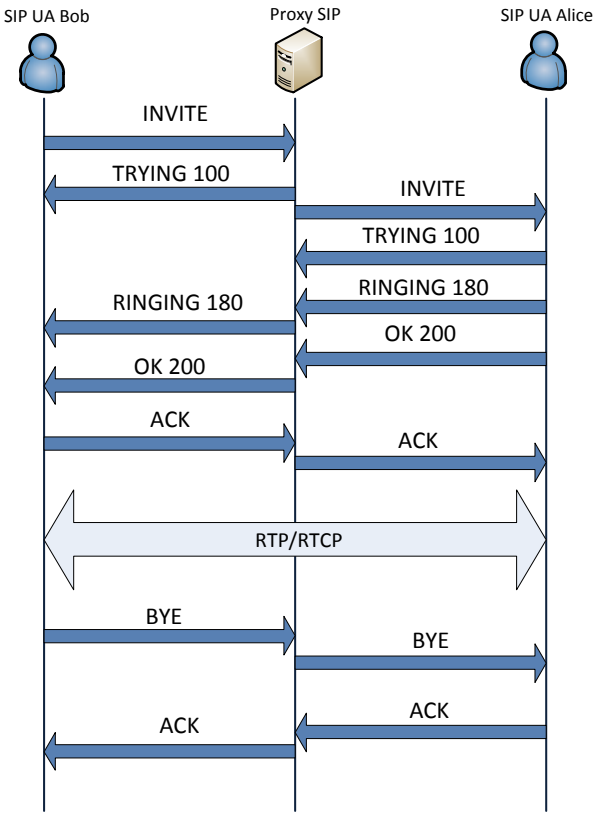

Ilustración 6 Ejemplo de comunicación SIP entre usuarios

En el ejemplo, se asume que ambos usuarios han realizado el registro de SIP correspondiente, previamente al establecimiento de la sesión multimedia. El proceso de establecimiento de sesión comienza con el envío de una solicitud INVITE desde el Agente de Usuario iniciador (UA de Bob en el ejemplo). Un posible ejemplo de una solicitud *INVITE* se muestra a continuación:

```
INVITE: sip:alice@dominio.com SIP/2.0
Via: SIP/2.0/UDP host.dominioAlice.com:5060
From: Bob <sip:bob@dominio.com>
To: Alice <alice@dominio.com>
Call ID: 764933@host.dominio.com
Cseq: 1 INVITE
Contact: sip:bob@dominio.com
Content-Type: application/sdp
Content-Length: 124
```

Mientras la petición *INVITE* llega a su destino, los servidores que dicha petición atraviesa generan respuestas temporales del tipo *100 TRYING*

Una vez el usuario destinatario de la petición recibe el *INVITE*, genera y envía, inmediatamente, un *180 RINGING* para indicar que se ha recibido el *INVITE*, y que ha comenzado el proceso de alerta del usuario destino (si éste es necesario). .En cuanto el usuario destinatario (Alice) acepta el establecimiento de la sesión, su Agente de Usuario correspondiente genera una respuesta de tipo *200 OK* que llega hasta el UA de Bob. Esta respuesta OK, puede ser similar a la siguiente:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP host.dominio.com
From: Bob <sip:bob@dominio.com>
To: Alice <alice@dominio.com>
Call ID: 764933@host.dominio.com
Cseq: 1 INVITE
Contact: sip:alice@dominio.com
Content-Type: application/sdp
Content-Length: 107
```

Finalmente, la sesión queda establecida con un *ACK* por parte del usuario que inició la sesión y a partir de ese momento puede comenzar el intercambio de datos mediante el uso de un protocolo de transporte, como por ejemplo, RTP. La sesión termina cuando uno de los usuarios envía una solicitud BYE (en este ejemplo, dicha solicitud es enviada por el UA de Bob).

2.2 *SDP*

SDP (*Session Description Protocol*), (6), es un formato utilizado para la descripción de sesiones multimedia establecidas entre dos o más usuarios, así como para la negociación de capacidades en una sesión. SDP comenzó como un componente de SAP (*Session Announcement Protocol*) (7), pero finalmente su mayor utilidad reside en la descripción de sesiones dentro del protocolo SIP y para la descripción de sesiones de tipo *multicast*.

Una descripción de sesión basada en SDP incluye información acerca de:

- Nombre de la sesión y su propósito
- Tiempo/os activo de la sesión
- Tipo de datos intercambiados en la sesión (vídeo, audio, etc.)
- Información necesaria para ser capaces de recibir dichos datos, como los puertos, direcciones, formatos, etc.
- Información adicional, como por ejemplo, el ancho de banda a utilizar en la sesión, o la información de contacto del usuario responsable de dicha sesión.

De esta forma, SDP proporciona toda la información suficiente y necesaria a todos aquellos usuarios que quieran participar en una sesión.

2.2.1 Contenido de SDP

El formato genérico de una descripción de sesión mediante el uso del protocolo SDP consiste en un conjunto de líneas, definido en la RFC (6) de la siguiente forma:

<type> = <value>

donde *type* es un identificador concreto del protocolo, y *value* es un texto estructurado que depende del valor tomado por *type*. La primera línea de la descripción de la sesión comienza por "v=". La descripción contiene un conjunto de secciones, que comienzan por "m=" y que permiten describir los componentes multimedia de la sesión (ej. componentes de audio o vídeo). Algunas líneas de la descripción son obligatorias, mientras que otras, marcadas con "*", son optativas:

Descripción de la sesión: información relativa a la sesión y al creador de la misma

v = versión del protocolo
o = identificador de la sesión y su origen
s = nombre de la sesión
i = información de la sesión*
u = URI de la descripción*
e = dirección de email*
p = número de teléfono*
c = información de la conexión*
b = información del ancho de banda*
z = ajustes de zonas horarias*
k = clave de encriptación*
a = atributos de sesión*

Descripción de tiempos: información de los tiempos de inicio y finalización, así como de las repeticiones de las sesiones.

t = tiempo que la sesión está activa.
r = tiempo de repetición*

Descripción del medio (multimedia): información sobre el protocolo de trasporte utilizado, o el tipo de medio soportado.

m = nombre del medio y dirección de transporte.
i = título del medio.*
c = información de conexión*
b = información de ancho de banda.*
k = clave de encriptación.*
a = atributos del medio.*

2.2.2 Formato de SDP

Un ejemplo de una descripción de sesión, obtenido de (6), se muestra a continuación:

```
v=0
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
s=SDP Seminar
i=ASeminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
```

Analizamos, más al detalle, cada uno de los campos.

Campo "*version*"

Almacena la version del protocolo SDP:

v=<version sdp>

Campo "*origin*"

Almacena información relativa al usuario que originó la sesión, así como los identificadores de dicha sesión, de forma que se pueda identificar la sesión de manera unívoca:

o=<username> <id sesion><version session><tipo red><tipo dirección IP><dirección IP>
o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5

- *username:* nombre del usuario que establece la sesión.
- *id de sesión*: identificador único de la sesión.

- *versión de sesión*: índica la versión de la sesión establecida.
- *tipo red*: indica el tipo de red por la que se desarrolla la sesión. Normalmente este valor es IN, que indica INTERNET.
- *tipo dirección IP*: indica el tipo de direccionamiento IP que se utiliza en la sesión (IPv4 o IPv6).
- *dirección IP*: es la dirección IP del usuario que estableció la sesión.

Campo "*session name*"

Indica el nombre de la sesion, y no puede ser un valor vacío:

s=SDP Seminar

Campo "*information*"

Indica información adicional acerca de la sesión que se quiere establecer o que se ha establecido:

i=ASeminar on the session description protocol

Campo "URI*"*

Indica la dirección donde se puede encontrar más información relativa a la sesión:

u=http://www.example.com/seminars/sdp.pdf

Campo "*email"*

Indica la dirección de correo electrónico del usuario que establece la sesión:

e=j.doe@example.com (Jane Doe)

Campo "*connection*"

Contiene información sobre el tipo de conexión:

c=<tipo de red><tipo direccion> <dirección conexion>

- *tipo de red*: ha de ser de tipo IN (INTERNET)
- *tipo dirección*: indica si es IPv4 o IPv6.
- *dirección conexión*: es la IP donde se reciben y desde donde se envían los datos. Esta IP puede ser de tipo *unicast*, o de tipo *multicast*, y no tiene por qué coincidir con la IP del mensaje *INVITE* en el que va incrustado la oferta SDP.

Campo "*bandwidth*"

Indica el ancho de banda que se ha propuesto para utilizar en la sesión:

b=<tipo bw>:<bw>

- *tipo bw*: puede tener dos posibles valores, AS que indica el ancho de banda relativo a una sola sesión, y CT que indica que es el ancho de banda repartido entre varias sesiones establecidas.
- *bw*: indica el ancho de banda en KB/seg (kilobytes).

Campo "*time"*

Indica los tiempos de inicio y finalización de la sesión:

t=2873397496 2873404696

Si los valores de tiempo son cero, implica que la sesión aún está en curso.

Campo "*attributes*"

Indica los atributos relativos tanto a la sesión, como a los datos intercambiados en la sesión. Por lo tanto, este parámetro, puede venir dado por dos formatos diferentes:

a=<atributo>

- *atributo = sendonly*: indica que en la sesión sólo se envía multimedia.
- *atributo = recvonly*: indica que en la sesión sólo se recibe multimedia.
- *atributo = sendrecv*: indica que en la sesión se envía y se recibe multimedia.
- *atributo = inactive*: indica que en la sesión no se quiere ni enviar ni recibir multimedia.

a=<atributo>:<valor>

En este caso, dentro de los atributos de valor, tenemos los relativos a calidad de servicio, como por ejemplo:

a=des:qos mandatory remote send

Indica que se desea (*des*) calidad de servicio obligatoria (*qos mandatory*) en los usuarios remotos, a la hora del envío (*remote send*)

O los que hacen referencia al mapeo de un formato de codificación determinado, como por ejemplo:

a=rtpmap:0 PCMU

Indica mapeo en el *payload* 0 con configuración tipo PCMU

Campo "*media"*

Indica el tipo de datos (multimedia) que son aceptados en la sesión.

m=<tipo media> <puerto> <tipo transporte> <lista formatos>

- *tipo media*: indica el tipo de datos que se intercambian en la sesión. Puede ser de tipo "*video*", "*audio*", "*text*", "*message*" y "*application*".
- *puerto*: indica el puerto por el que se envían/reciben los datos multimedia.
- *tipo transporte:* indica el protocolo de transporte que se usa en la sesión para los datos multimedia. Puede tomar como valor "UDP" cuando no se

especifica ninguno en concreto, o por ejemplo, "RTP/AVP" en el que se establece el protocolo RTP para el transporte del vídeo o el audio, o "RTP/SAVP", en el que la "S" indica modo seguro.

- *lista formatos*: indica los formatos de codificación soportados.

A modo de ejemplo, se proporcionan las siguientes descripciones "m=":

m=audio 49170 RTP/AVP 0
m=video 51372 RTP/SAVP 99

2.2.3 Modelo oferta/respuesta de SDP

Este modelo de oferta/respuesta fue definido en la RFC (8) y describe cómo ha de establecerse la descripción de una sesión entre dos usuarios. Según este modelo, un usuario, el ofertante, genera una oferta SDP con sus preferencias respecto a la sesión multimedia que quiere establecerse. Otro usuario, el ofertado, recibe la oferta SDP y puede, o bien rechazarla o bien aceptarla. Al aceptarla, deberá generar su propia carga SDP y reenviársela de vuelta al usuario inicial. Esta respuesta debe contener el mismo número de líneas *m* que se recibieron en la oferta inicial.

En la Ilustración 7 se muestra un ejemplo de este proceso. En la oferta inicial generada por el usuario Alice, se caracteriza el establecimiento de una sesión multimedia basada en una componente de audio y dos componentes de vídeo, indicando los formatos H.261 y MPEG soportados. En la respuesta generada por Bob, se observa como únicamente se acepta una de las peticiones de vídeo.

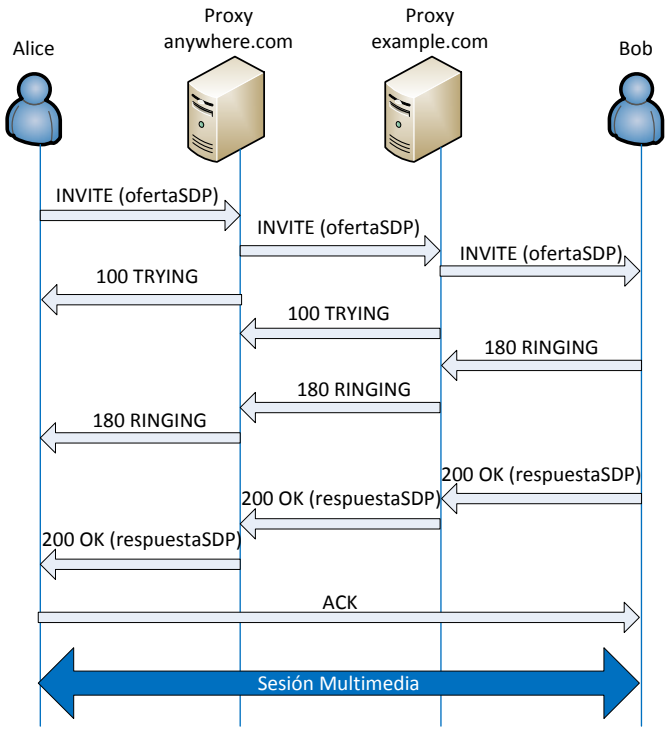

Ilustración 7 Modelo oferta/respuesta de SDP (8)

Oferta de Alice

```
v=0
o=alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=
c=IN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31
a=rtpmap:31 H261/90000
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

Respuesta de Bob

```
v=0
o=bob 2890844730 2890844730 IN IP4 host.example.com
s=
c=IN IP4 host.example.com
t=0 0
m=audio 49920 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 0 RTP/AVP 31
m=video 53000 RTP/AVP 32
a=rtpmap:32 MPV/90000
```

2.3 *IMS*

IMS (*IP Multimedia Subsystem*) es una arquitectura de red que permite integrar servicios multimedia sobre tecnología IP. Estos servicios multimedia van desde la voz sobre IP (*VoIP*), pasando por el *Video Streaming*, el *Push To Talk* (*PTT*), la Videoconferencia, etc… (9). Es un estándar desarrollado por el 3GGP (10) junto con el IETF (5).

Existe un concepto, denominado "*ALL-IP*" que se basa en que las diferentes tecnologías de acceso se puedan integrar de una manera sencilla y transparente en una capa de red basada en el protocolo IP, de forma que, además de ganar en sencillez y eficacia, se proporciona a los operadores beneficios como el ahorro de costes por infraestructuras, escalabilidad, sencillez en las operaciones y el mantenimiento de la red, etc. IMS implica la aplicación de este concepto, "*ALL-IP*", en las redes 3G.

Por tanto, IMS soporta servicios con diferentes tecnologías de acceso, como por ejemplo, GSM, GPRS, UMTS, *Wi-Fi*, *Bluetooth*, etc. De esta forma, IMS hace converger las comunicaciones de telefonía junto con las de Internet, de forma que los usuarios puedan utilizar indistintamente ambos servicios, o combinarlos entre ellos, dependiendo de sus necesidades.

Se puede pensar, que tanto la telefonía como Internet, están perfectamente adaptados a sus respectivas redes como para dar los servicios que los usuarios demandan, por tanto, ¿para qué necesitamos IMS?, pues principalmente por los siguientes tres motivos, según se especifica en (9):

- *QoS* (*Quality of service*): la telefonía móvil, basada en redes de conmutación de paquetes, se caracteriza por aplicar QoS de tipo "*best-effort*", lo que significa que la red no garantiza nada acerca del ancho de banda que un usuario obtiene para una comunicación, ni la calidad de la llamada, lo que choca directamente con las características que debe tener hoy en día los proveedores de servicios multimedia y de tiempo real. Es por esto que IMS adquiere tanta importancia, ya que es capaz de proveer, para cada comunicación establecida, (ya sea de VoIP, descarga de vídeo, o una simple comunicación telefónica) una calidad de servicio determinada y adaptada a las necesidades del cliente.

- *Facturación (acceso/servicio/contenido):* a consecuencia del *QoS*, los proveedores de estos servicios son capaces de ofrecer a sus clientes diferentes

calidades en sus comunicaciones, y en consecuencia, realizar diferentes cargos según el servicio que contraten sus clientes, basándose tanto en el tipo de acceso, el servicio o el contenido. Sin embargo, no es IMS quien fija cómo ha de realizarse el cargo al cliente, sino que lo que hace es facilitar toda la información relativa del cliente y su sesión de comunicación establecida, para dejar en manos del proveedor el cargo.

- *Integración de diferentes servicios*: cada vez es mayor la oferta de servicios multimedia ofrecidos a los usuarios por las diferentes empresas de comunicaciones, de forma que es necesario que los proveedores de estos servicios, como son las grandes compañías de telefonía, sean capaces de integrarlos todos en un único servicio para sus clientes. Por ello, IMS define las interfaces estándares a ser utilizadas por los desarrolladores de estos servicios y que, de esta forma, los proveedores puedan ofrecer servicios mucho más potentes y de diferentes suministradores, aumentando de esta manera su oferta y calidad.

Por tanto, lo que hay que tener claro de IMS es que, no define en ningún caso las aplicaciones finales, sino las infraestructuras y bloques necesarios para proveer de los servicios demandados.

2.3.1 Planos Lógicos de IMS

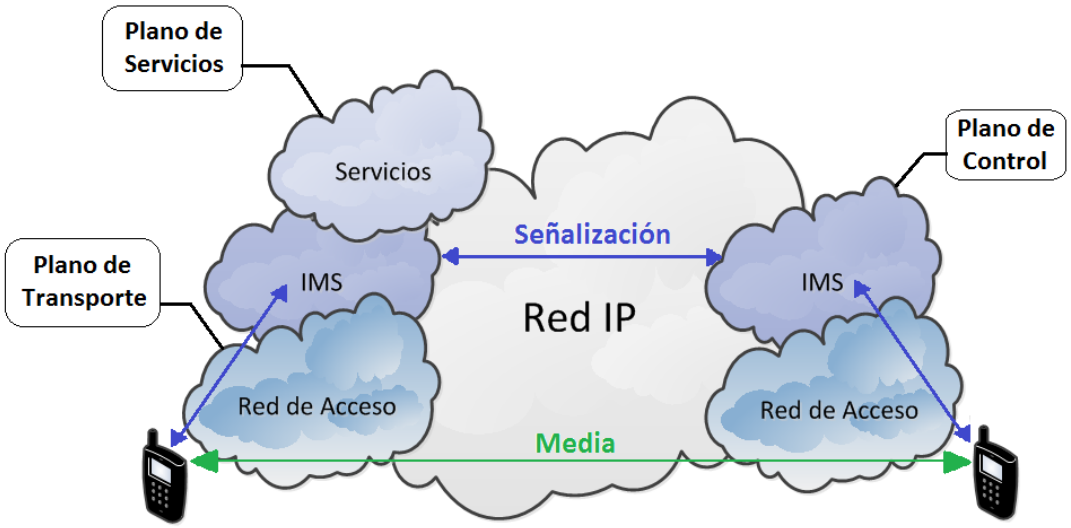

Ilustración 8 Planos lógicos de IMS (11)

En IMS se implementa el plano de control para los servicios multimedia, mientras que el plano de transporte es implementado según los estándares definidos, como GPRS/3G, o cualquier otro, como por ejemplo *WiFi*, etc. Finalmente, los datos, no pasan en ningún caso, a través de la red IMS.

2.3.2 Arquitectura IMS

Los elementos que forman la red IMS pueden clasificarse según su funcionalidad en:

- Elementos de control: S-CSCF, I-CSCF y P-CSCF.
- Elementos de almacenamiento de datos: HSS y SLF.
- Servidores de aplicación y pasarelas: SIP AS, OSA-SCS y IM-SSF.
- Funcionalidad Multimedia: MRFC y MRFP.
- Funcionalidad de *Internetworking*: MGW/MGCF, SGW y BGCF.

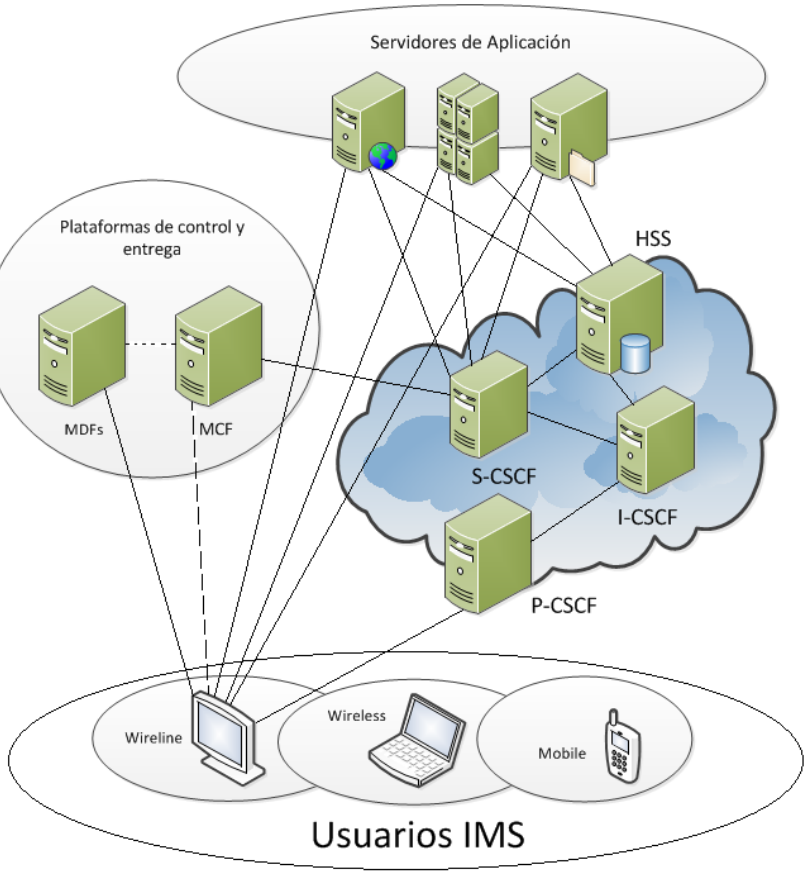

Ilustración 9 Arquitectura de IMS

A continuación, se describen más en detalle las funcionalidades de los elementos que forman la red IMS. En particular, la descripción proporcionada a continuación se centra en los elementos de control (CSCFs), de almacenamiento de datos (HSS y SLF) y en los servidores de aplicación.

2.3.3 C-SCF (*Call Session Control Function*)

Estos elementos actúan como nodos de control para la señalización SIP, y por lo tanto, son los elementos básicos para proporcionar funcionalidad en la red IMS. Pueden ser de tres tipos diferentes:

- Proxy-CSCF (P-CSCF): es el punto de entrada que establece el primer contacto entre el usuario IMS y la propia red, y por tanto, actúa como un servidor proxy para el terminal.

- Serving-CSCF (S-CSCF): se encarga de proveer de control a la sesión, de manera que a cada usuario se le asigna un elemento S-CSCF durante el proceso de registro, que es el encargado después, de su autenticación, así como de aplicarle las políticas de sesión que tenga asignado dicho usuario. Por ello, se comunica directamente con el HSS y el SLF que son los encargados de suministrarle toda la información que necesite relativa al usuario.

- Interrogating-CSCF (I-CSCF)*:* es el punto de entrada a la red IMS desde otras redes. Actúa como un servidor SIP Proxy, de manera que oculta la topología de la red IMS hacia las redes externas. Realiza también tareas de autenticación de usuarios, por lo que se conecta directamente con el HSS y el SLF.

2.3.4 HSS (*Home Subscriber Server*)

Es el elemento encargado del almacenamiento de toda la información relativa al usuario. Esta información consiste tanto en datos de autenticación, como en datos de calidad de servicio, políticas de perfil de usuario, seguridad, identidad de usuarios, etc.

Dentro de las identidades de usuario, podemos encontrar dos tipos: las públicas y las privadas. Las identidades privadas se utilizan para los procesos de autenticación y registro de los usuarios, mientras que las identidades públicas son aquellas que los usuarios utilizan para establecer las conexiones, son el "*alias*" de cada usuario.

Además, el HSS almacena información en los IFC (*Initial Filter Criteria*), que son los encargados de realizar un filtrado de la señalización, de manera que los mensajes SIP son encaminados al servidor de aplicaciones correspondiente al servicio establecido para el usuario.

2.3.5 SLF (*Subscription Location Function*)

Elemento utilizado cuando la red IMS dispone de más de un elemento HSS, actuando como un redirector de peticiones hacia el HSS que corresponda.

2.3.6 Servidores de Aplicación y Pasarelas

Son los elementos donde se implementan los servicios y están encuadrados dentro del plano de aplicación. Se utilizan para poder proporcionar servicios de valor añadido a la red y a sus usuarios.

Según la definición del 3GPP, en el plano de servicios la señalización puede ser tratada de una manera u otra, según una serie de criterios que quedan definidos en el

bucle de abonado y almacenados en el HSS, de manera que el S-CSCF se descarga dicha información y se encarga de enrutar la señalización hacia los servidores de aplicaciones (SIP), o pasarelas, como por ejemplo, OSA (OSA SCS – *Open Service Architecture Service Capability Server*) o CAMEL (*Customized Applications for Mobile network Enhaced Logic*).

2.3.7 Principales protocolos utilizados en IMS

Dentro de SIP, cabe destacar los principales protocolos que utiliza la red para su funcionamiento. Estos protocolos son SIP (4), Diameter (12) y RTP/RTCP (13), todos ellos desarrollados por el IETF (5).

El protocolo SIP (ver apartado 2.1) es utilizado en la red IMS en el plano de señalización para el establecimiento de las sesiones multimedia. El protocolo RTP/RTCP (ver apartados 2.5 y 2.6) es utilizados en el plano de datos, para el transporte de la información multimedia. Finalmente, el protocolo Diameter es utulizado en IMS para proporcionar las funcionalidades relacionadas con autenticación, autorización y Facturación (AAA, *Authentication, Authorization and Accounting*).

2.4 *Vídeo bajo Demanda (VoD)*

El vídeo bajo demanda, o *Video on Demand* (VoD), se basa en la idea de un cliente, que a través de un dispositivo, como una *Tablet*, un pc o un televisor con una set-top-box, se conecta a un servidor de vídeo, desde el cual obtiene una película y comienza a reproducirla, además de poder, en cualquier momento, pausarlo, pararlo o volverlo a reproducir.

A nivel técnico, el VoD nos permite realizar las mismas acciones que realizaríamos, por ejemplo, con un DVD, pero con la ventaja de que el vídeo se encuentra almacenado en remoto en un servidor.

Para obtener ese flujo de datos desde el servidor de vídeo, el cliente ha de realizar una petición especificando qué vídeo quiere visualizar y cómo lo quiere recibir (*codecs*, puerto de recepción, dirección IP…). Cuando el vídeo comienza a ser enviado al cliente, desde el servidor, éste lo empieza a almacenar en un *buffer*, que cuando alcanza un porcentaje de llenado, permite al cliente comenzar a visualizarlo. Esta técnica de visualización de un vídeo en el cliente, se conoce como s*treaming* y proporciona mucha mejora en términos de retardos y cortes en la visualización.

Lo más importante del VoD con respecto a otros elementos que nos proporcionan la visualización de contenidos, como por ejemplo, una televisión convencional, es que el VoD nos permite seleccionar lo que queremos ver, cuándo lo queremos ver y cómo lo queremos ver.

2.4.1 Requisitos del VoD

Los principales requisitos que ha de cumplir cualquier sistema de VoD, son los siguientes (14):

- **Gran capacidad de almacenamiento:** el contenido multimedia puede llegar a ocupar gran espacio en memoria, por lo que necesitaremos servidores de vídeo que sean capaces de almacenar gran cantidad de contenido.

- **Suficiente ancho de banda:** un sistema de vídeo de este tipo tiene que ser capaz de servir a múltiples clientes que le realicen peticiones de vídeos diferentes, y para todos ellos ha de cumplir las condiciones de retardo y calidad esperadas, por lo que necesita disponer de un ancho de banda elevado.

- **Servicio en tiempo real:** en este tipo de sistemas, el cliente tiene la capacidad de controlar la reproducción del flujo de datos a su antojo y bajo la premisa de hacerlo en tiempo real, por lo que este tiempo de respuesta desde que, por ejemplo, el cliente realiza una petición de pausa del vídeo, hasta que la reproducción se pausa, ha de ser mínimo e imperceptible por el cliente.

- **Calidad de servicio:** este tipo de sistemas deben producir el mínimo número de retardos, cortes o distorsiones en los datos, para garantizar al cliente una calidad de servicio óptima. Este término está íntimamente relacionado con el ancho de banda, debido a que dos clientes diferentes, con distinta capacidad de ancho de banda, pueden querer acceder al mismo contenido de vídeo, pero para que el cliente con menor ancho de banda reciba su vídeo en óptimas condiciones de calidad, es necesario que el propio servidor mantenga almacenados diferentes formatos para un mismo vídeo (gran capacidad de almacenamiento), de forma que pueda servir adecuadamente a cada cliente, dependiendo de las características específicas del mismo.

2.4.2 Arquitecturas del VoD:

Dentro de los sistemas de vídeo bajo demanda, podemos distinguir tres tipos de arquitecturas básicas (14):

- **Arquitectura centralizada**: el servidor de vídeo es un sistema centralizado al cual se conectan los clientes a través de la red.

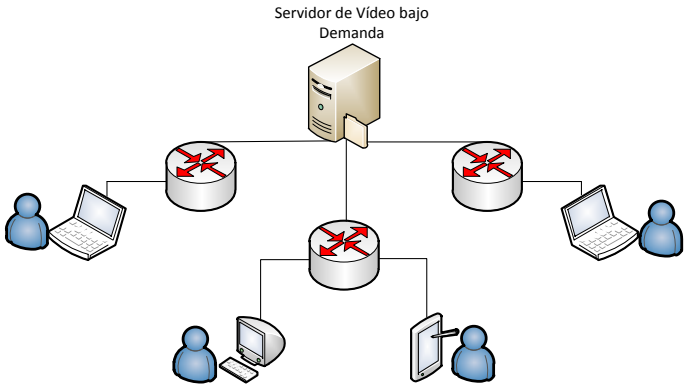

Ilustración 10 Arquitectura centralizada de vídeo bajo demanda

- **Arquitectura basada en un servidor independiente o proxy**: los usuarios realizan sus peticiones de vídeo a servidores intermedios, o proxies, de manera que son los propios proxies los que establecen conexión directa con el servidor de vídeo.

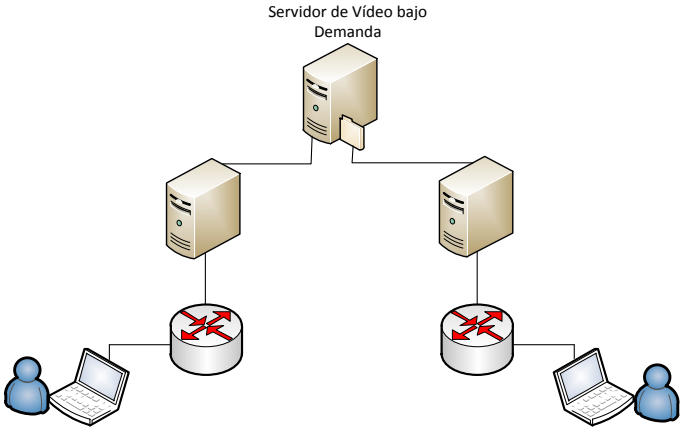

Ilustración 11 Servidor de vídeo bajo demanda basado en servidores proxy

- **Arquitectura distribuida**: o modelo *peer-to-peer*, en el que el propio cliente se convierte a su vez en servidor del contenido bajo demanda.

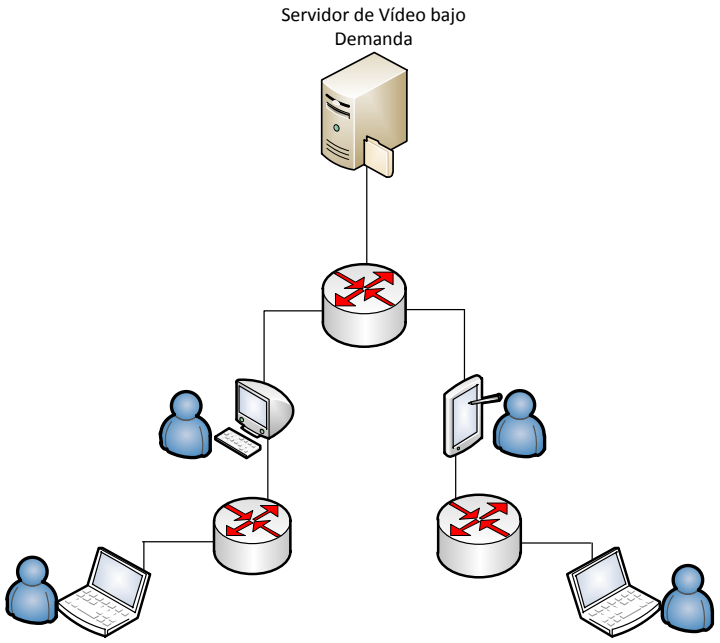

Ilustración 12 Servidor de vídeo bajo demanda basado en un sistema distribuido

2.4.3 Especificación de TISPAN para VoD

El grupo TISPAN (*Telecommunications and Internet converged Services and Protocols for Advanced Networking*) de ETSI (*European Telecommunications Standards Institute*) es un organismo de estandarización internacional, encargado, entre otras, de realizar las especificaciones técnicas sobre la arquitectura y funcionalidad de sistemas IPTV (*IP Television*). Estos sistemas se establecen sobre modelos de redes de nueva generación (NGN, *Next Generation Networks*), por las capacidades de transporte y mecanismos de calidad de servicio que ofrecen.

En la versión 2 de especificaciones de TISPAN (3) se define un sistema de IPTV que hace uso de las funcionalidades de control ofrecidas por IMS. Dicho sistema ofrece tres servicios principales: *Broadcast TV*, *Network-Personal Video Recorder* (N-PVR) y *Content on Demand* (CoD). Éste último recoge la definición de un servicio de Vídeo bajo Demanda de próxima Generación basado en la arquitectura de IMS.

En la versión 3 de especificaciones de TISPAN, se añade un amplio abanico de servicios de valor añadido. Algunos ejemplos son televisión interactiva, soporte de contenido generado por el usuario, perfiles de usuario y personalización, comunicaciones y mensajería, notificaciones, etc.

En la Ilustración 13 se muestra la arquitectura del sistema IPTV desarrollado por TISPAN, el cual soporta el servicio de Vídeo bajo Demanda en redes con plano de control IMS. La especificación de dicho sistema se recoge en los documentos técnicos (2) y (3).

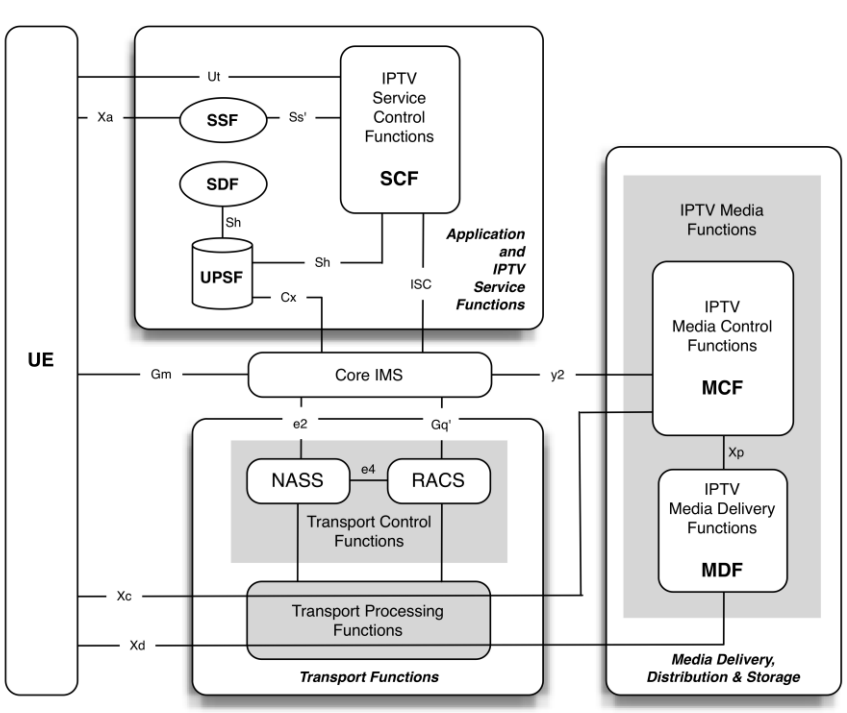

Ilustración 13 Especificación TISPAN para sistemas IPTV basados en IMS (3)

En la especificación 3 del mismo documento, TISPAN define 3 servicios principales para el desarrollo de sistemas IPTV basados en IMS: *Broadcast TV*, *Network-Personal Video Recorder* (N-PVR) y *Content on Demand* (CoD). Se incluyen, además, especificaciones par servicios de valor añadido, como la televisión interactiva o los canales personalizados. Todas estas especificaciones se pueden encontrar en (3) y (15).

2.5 *Real Time Protocol (RTP)*

RTP (*Real Time Protocol*) es un protocolo de nivel de transporte definido por el IETF (13), que proporciona un servicio de entrega extremo a extremo para datos con características en tiempo real, por ejemplo audio y vídeo. El servicio proporcionado permite a una aplicación usuaria de RTP detectar pérdidas de paquetes y paquetes entregados fuera de secuencia, mediante el use de números de secuencia en el nivel de transporte. Asimismo, RTP transporta marcas de tiempo en la cabecera de los paquetes de datos multimedia, lo que posibilita la implementación a nivel de aplicación de mecanismos para evitar los efectos del Jitter, y permite reproducir el contenido multimedia de cada paquete en el instante apropiado (por ejemplo, permite reproducir flujos de audio y vídeo sincronizados). Finalmente, RTP permite transportar datos multimedia con distintas codificaciones, propietarias o estandarizadas, identificando el formato empleado en la cabecera RTP de cada paquete.

El protocolo funciona típicamente sobre UDP, si bien el uso de otros protocolos orientados a conexión (ej. TCP) es posible, y soporta el envío de información multimedia a múltiples destinatarios mediante el uso de tecnologías de entrega multicast. Finalmente, es importante destacar que RTP no proporciona QoS, fiabilidad o entrega ordenada de paquetes. Estas dos últimas funcionalidades pueden ser implementadas a nivel de aplicación, teniendo en cuenta la información transportada en los números de secuencia de RTP.

2.6 *Real Time Streaming Protocol (RTSP)*

RTSP (*Real Time Streaming Protocol*) es el protocolo encargado de controlar flujos de datos multimedia entre un cliente y el servidor de dichos datos. Se puede decir, que el protocolo RTSP realiza las funciones de "mando a distancia" hacia el servidor multimedia que está emitiendo de manera continua dicho flujo de datos (16).

El conjunto de *streams* sujetos al control por parte de RTSP quedan definidos en una descripción de sesión establecida previamente mediante el uso de protocolos como SIP y SDP.

Por lo general, RTSP utiliza dos protocolos de transporte diferentes, TCP para los datos de control del *streaming* y RTP para los datos multimedia propiamente dichos, aunque para este tipo de datos, si la red lo requiere, también puede utilizarse TCP. El cliente puede realizar tantas peticiones de flujos de datos como desee, tanto con el protocolo UDP como con el protocolo TCP.

Las principales características del protocolo RTSP se describen a continuación (16):

- **Extensible:** implica que ante la aparición o desarrollo de nuevos métodos y parámetros para el protocolo, pueden ser aplicados a éste de manera sencilla.

- **Seguro:** utiliza mecanismos de seguridad web que pueden ser aplicados tanto al nivel de transporte, como a nivel del propio protocolo.
- **Independiente del protocolo de transporte:** tal y como se comentó con anterioridad, es capaz de usar tanto UDP, como TCP, dependiendo de los requisitos de la red.
- **Multiservidor:** el cliente tiene la posibilidad de pedir el mismo flujo multimedia a diferentes servidores que lo contengan, de forma que se establecen sesiones concurrentes del control de dichos *streamings*.

2.6.1 Las peticiones RTSP

Las peticiones del protocolo RTSP se envían directamente desde el cliente hasta el servidor. Su formato es muy similar al establecido para las peticiones de tipo HTTP a los servidores web.

Nombre	Uso	Descripción
DESCRIBE	Recomendado	Facilita la información relativa a la sesión establecida para la URL que corresponde con el servidor. Debe contener todos los datos de inicialización relativos a la sesión que describe.
ANNOUNCE	Opcional	Cuando es de cliente a servidor indica el anuncio de la descripción de sesión que se va a establecer. Cuando es de servidor a cliente se anuncia dicha sesión en tiempo real, realizando una actualización de la sesión.
GET_PARAMETER	Opcional	Realiza la petición de un parámetro determinado de la sesión asociada al flujo de datos que llega desde el servidor. Cuando este comando no lleva un parámetro en concreto, sino que va vacío, se utiliza para la realización de un ping hacia el servidor, para comprobar que sigue vivo.
OPTIONS	Requerido	Cuando el cliente envía esta petición al servidor, éste le contesta con una información en la que puede incluirse la versión RTSP, el número de sesión, la fecha o el nombre del servidor.
PAUSE	Recomendado	Indica al servidor la interrupción momentánea de los datos

		multimedia que se están sirviendo
PLAY	Requerido	Indica al servidor el comienzo del envío de los datos multimedia, según los mecanismos de transporte establecidos en *SETUP*.
RECORD	Opcional	Inicia la grabación de una parte de los datos multimedia transmitidos. Puede indicarse un tiempo de grabación, o si no se grabará todo el flujo de datos
REDIRECT	Opcional	Le indica al cliente que debe conectarse a otro servidor para continuar con la recepción de los datos multimedia. De esta forma, cuando el cliente recibe este tipo de petición, manda una petición de tipo *TEARDOWN* a su servidor actual, y una petición *SETUP* al nuevo servidor.
SETUP	Requerido	Indica el tipo de mecanismo de transporte que se utiliza en la sesión. La petición *SETUP* puede enviarse de nuevo, una vez la sesión esté establecida, para cambiar los parámetros del transporte.
SET_PARAMETER	Opcional	Realiza la petición de ajuste de un parámetro determinado de la sesión asociada al flujo de datos que se está transmitiendo desde el servidor. Sólo se permite el cambio de un parámetro cada petición de *SET_PARAMETER*.
TEARDOWN	Requerido	Indica al servidor la parada inmediata de los datos que está sirviendo, liberando todos los recursos asociados a la sesión. Hasta que no se establezca una nueva sesión de *SETUP*, no podrá restablecerse la sesión.

En la siguiente figura puede observarse un escenario básico de intercambio de mensajes de control RTSP entre un cliente, y un servidor:

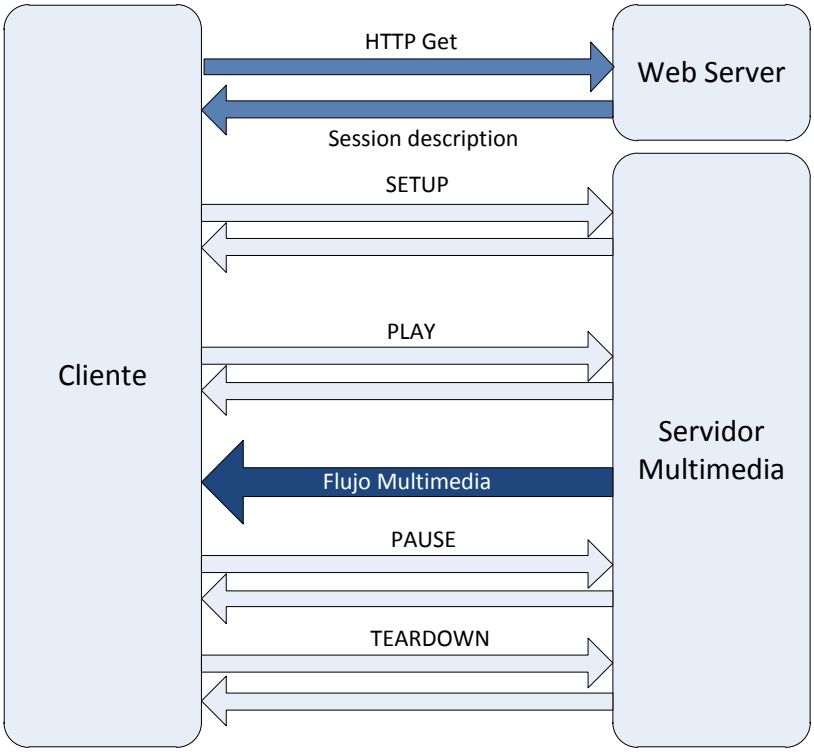

Ilustración 14 Intercambio de peticiones RTSP entre cliente y servidor multimedia

En la figura puede observarse como el cliente, establece una descripción de la sesión con el servidor web, (en el intercambio HTTP inicial), y a continuación se establece el intercambio de mensajes de tipo RTSP para el control de los datos multimedia que se envían desde el servidor multimedia hasta el cliente. Tras el envío del mensaje RTSP PLAY, el servidor responde con un mensaje OK de RTSP y comienza el envío del contenido multimedia hacia el cliente.

2.7 *Web RTC*

Este capítulo del estado del arte se desarrolla porque inicialmente, este proyecto fue concebido con una premisa inicial, que fue la utilización de la tecnología WebRTC para la implementación del mismo. Concretamente, esta tecnología iba a ser utilizada para desarrollar la interfaz Web del cliente, utilizando el soporte que proporciona para la gestión de cargas SDP y de reproducción de contenido multimedia.

Al tratarse de una tecnología bastante nueva, y tras unos meses de estudio y desarrollo, concluimos que en ese momento esta tecnología no podía ser usada de manera sencilla para los fines de implementación de este proyecto.

2.7.1 ¿Qué es WebRTC?

WebRTC (*Web Real Time Communications*) surge como una herramienta para la implementación de la necesidad, cada vez más extendida, de comunicación en tiempo real a través de dispositivos electrónicos con conexión a Internet, de una manera más rápida y eficaz.

Es un proyecto de código abierto desarrollado por *Google* y a partir de los estándares definidos por el W3C (*World Wide Web Consortium)* y el IETF, que se basa en permitir a los desarrolladores web la capacidad de desarrollar aplicaciones multimedia (como por ejemplo, vídeo o chat), sobre *browsers*, con capacidad de comunicación en tiempo real y sin necesidad de descarga de ningún tipo de *pluggin* o aplicación adicional (17). Esto permite la disminución de graves problemas que han conformado siempre las aplicaciones multimedia tipo conferencia, o intercambio de archivos, como son, la seguridad, o los *codecs*, entre otros. Así, al ser un estándar soportado por la gran mayoría de *browsers* (*Chrome*, *Chrome Canary*, *Firefox*, *Mozilla*, *Opera*...), se pueden establecer comunicaciones entre ellos, permitiendo así, la fácil conexión entre diferentes dispositivos, como móviles, *tablets* y portátiles.

En la siguiente frase, queda plasmado perfectamente el fundamento principal de webRTC (17):

"WebRTC and HTML5 could enable the same transformation for real time that the original browser did for information"

Phil Edham
NoJitter

2.7.2 Comunicaciones en tiempo real:

Actualmente, todas las aplicaciones basadas en comunicaciones en tiempo real están formadas por tres elementos principales: un *framework* (dispositivo/sistema operativo del usuario); una interfaz de usuario; y una *media engine*, encargada de lidiar con la transmisión/recepción de los archivos utilizados en dicha comunicación.

La *media engine* es la encargada, principalmente, de llevar a cabo las funciones que permiten entregar audio y vídeo con la calidad esperada (17):

- Audio: control hardware, RTP, encriptación, baja latencia en la recepción del audio y vídeo, reducción del ruido, manejo de los *codecs*, tratamiento de pérdidas…
- Vídeo: reproducción y captura por cámara del vídeo, procesamiento del vídeo, administración del ancho de banda, manejo de *codecs*, reducción de ruido…

WebRTC se hace cargo de todas las funciones realizadas por la *media engine*, ayudándose de una serie de estándares y APIs, para así crear una nueva solución a las comunicaciones en tiempo real basada en la utilización directa de aplicaciones sobre *browsers*.

Además de las funciones básicas de la comunicación, como los codecs, WebRTC incluye una API (*Application Programming Interface*) que permite controlar el software del servidor y así conseguir una conexión directa entre dos dispositivos de WebRTC sin necesidad de ningún otro tipo de señalización o protocolo externo, una vez dicha comunicación haya sido establecida. Las tres APIs que forman parte de esta tecnología

2.7.3 Establecimiento de la conexión en WebRTC:

El establecimiento de conexión se basa en un modelo desarrollado por *Google* tras la definición de WebRTC, denominado "aproximación JSEP (*Javascript Session Establishment Protocol*)", en el que se estableció el plano de señalización y el plano de los datos.

De esta manera, en el *browser* se implementa lo mínimo posible (como *codecs* y encriptación), y dejamos al nivel de aplicación implementar el resto. El nivel de aplicación se encarga de entregarle al *browser* los parámetros necesarios para el establecimiento de la conexión, mediante el uso de los descriptores de sesión, o *session descriptors*. Formará parte también, del nivel de aplicación de realizar el intercambio de información con el otro lado de la comunicación de la manera que quiera, como por ejemplo, utilizando servidor *AppEngine* o servidor SIP:

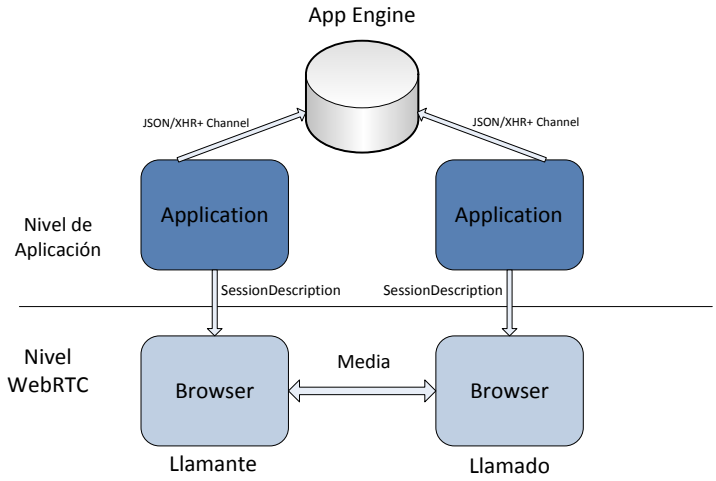

Ilustración 15 Implementación de WebRTC mediante AppEngine (17)

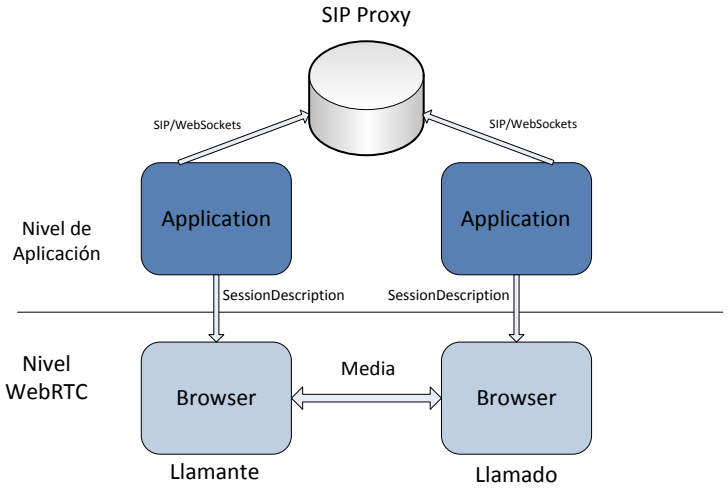

Ilustración 16 Implementación de WebRTC mediante SIP Proxy (17)

Para el comienzo de una sesión, lo que el cliente necesita es:

- Descripción de la sesión local: describe la configuración del lado local
- Descripción de la sesión remota: describe la configuración del lado remoto
- Candidato/os de transporte remotos: describe cómo conectarse al lado remoto, es decir, direcciones de puertos e IPs, donde se encuentra el usuario remoto.

Tras la definición de las bases del establecimiento de la comunicación, las implementaciones de ésta pueden ser descritas, principalmente, a través de dos figuras geométricas, que son el triángulo y el trapezoide.

La comunicación más típica que se puede establecer, consiste en una comunicación en triángulo como la mostrada en la siguiente figura:

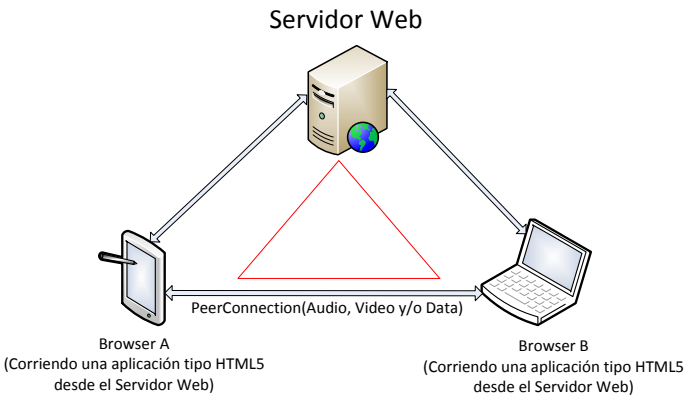

Ilustración 17 Comunicación WebRTC en triángulo (18)

Donde ambos *browsers* corren la misma aplicación WebRTC, a través de la misma página web. El intercambio de los parámetros de la sesión se establece entre los *browsers* y el servidor web, mientras que el intercambio de datos se produce de manera directa a través de una *PeerConnection* directa entre *browsers*. La señalización puede correr sobre HTTP o WebSockets hacia el mismo servidor HTML que sirve las páginas web al *browser*, o hacia otro servidor independiente que simplemente trate dicha señalización.

Para el caso de la comunicación tipo trapezoide, la principal diferencia es el uso de dos servidores, tal y como muestra la siguiente figura:

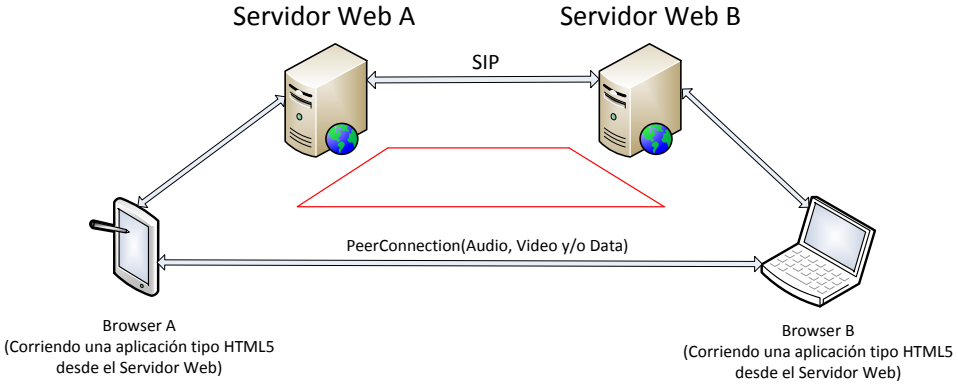

Ilustración 18 Comunicación WebRTC en trapezoide (18)

En este caso, se establece un mecanismo de señalización (SIP, o Jingle, por ejemplo), entre ambos servidores para facilitar el intercambio de los datos de la conexión. Esta configuración permite la interconexión de sistemas implementados por distintos suministradores, utilizando WebRTC como estándar.

Finalmente, cabe destacar que WebRTC está referido, normalmente, a aplicaciones *peer-to-peer*, que no son lo mismo que aplicaciones *browser-to-browser*. De hecho, según la propia especificación del protocolo, no es necesario que un dispositivo tenga un *browser* para poder utilizarlo, ya que tiene la capacidad de ser usado en cualquier tipo de dispositivos, como por ejemplo, una televisión (tele presencia), o un

coche (estado de las carreteras). Es aquí, por tanto, donde reside el gran potencial que una herramienta como esta puede ofrecernos, basándonos en el crecimiento exponencial en el uso de *smartphones* y *tablets*, así como las nuevas tecnologías y las redes de cuarta generación (4G).

2.8 *Jain SIP*

JAIN SIP (19) (20), es un conjunto de APIs de Java desarrolladas para la implementación de aplicaciones que usan el protocolo SIP como protocolo de señalización. Es decir, establecen la pila SIP y las solicitudes (peticiones y respuestas) necesarias para utilizar el protocolo SIP en la aplicación Java que se esté desarrollando.

La especificación de JAIN SIP define:

- La interfaz de la pila: definición de la pila SIP a partir de la cual se generarán las peticiones y las respuestas del protocolo.
- La interfaz de los mensajes: definición de las peticiones y respuestas del protocolo SIP.
- Los eventos: definición de los eventos que se encargan de informar cuando ha llegado una respuesta o una petición SIP.

Estas APIs, utilizadas para la implementación y el desarrollo del protocolo SIP en el ámbito de las aplicaciones, se basan en un modelo asíncrono de eventos del tipo *Listeners/Providers*, de manera que se asigna a la aplicación la capacidad de tratar con el envío y la recepción de mensajes del protocolo SIP. Este modelo asíncrono utiliza identificadores para agrupar los mensajes utilizados en dos grupos: diálogos y transacciones.

Una transacción está formada por una petición y su consecuente respuesta, mientras que un diálogo es el contexto en el que han de interpretarse todos los mensajes, de manera que, en un mismo diálogo hay una o más transacciones.

La estructura de JAIN SIP está integrada por los siguientes elementos:

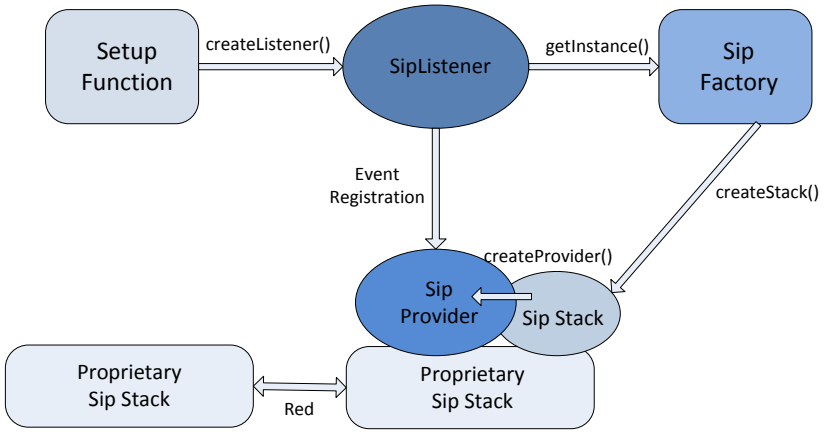

Ilustración 19 Arquitectura Jain SIP (20)

- **Interfaz SIPStack**: es la pila SIP de la aplicación a partir de la cual se podrán establecer las peticiones y respuestas que la aplicación requiera. Entre sus propiedades se encuentran, además de tener una dirección IP, el nombre de la pila, su proxy de salida o filtros de retransmisión.

- **Interfaz SIPProvider:** representa la entidad de mensajes de una pila SIP, de manera que se encarga de generar y enviar las peticiones SIP y de recibir y procesar las respuestas SIP.

- **Interfaz SIPListener:** se encarga de procesar los eventos que ocurren cada vez que se envía una petición o se recibe una respuesta, de manera que controla, por ejemplo, los *timeouts* (fin de la espera a una respuesta), o los tiempos de retransmisión de ciertas peticiones (notificaciones).

Cada una de las interfaces anteriormente definidas, engloba un entorno diferente dentro del ámbito del protocolo SIP. Esos entornos, se muestran en la siguiente figura:

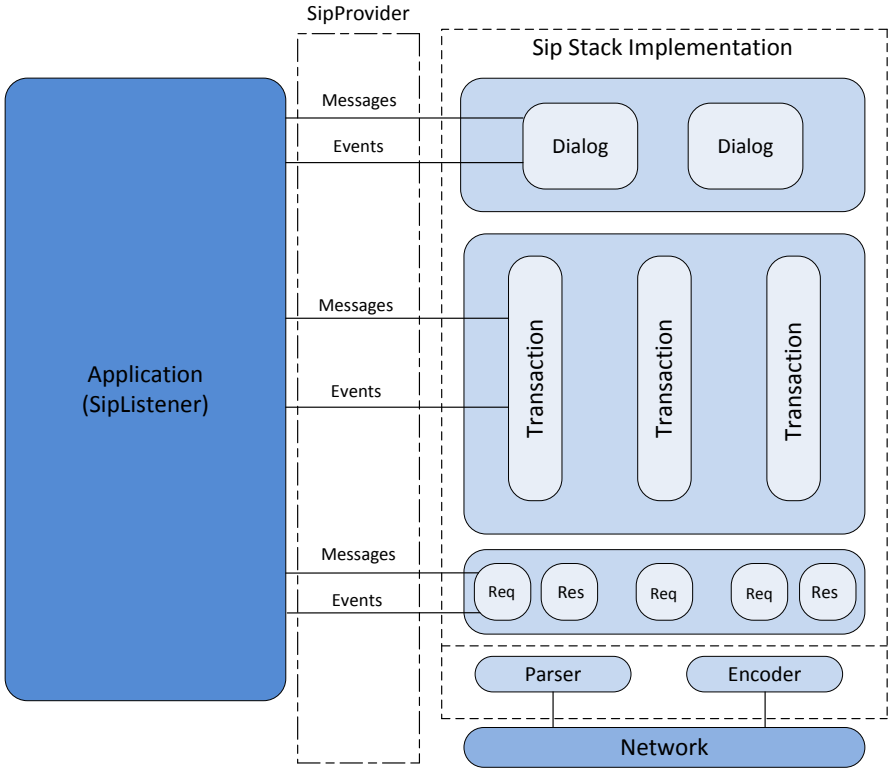

Ilustración 20 Estructura de la implementación Jain SIP (20)

La interactuación de todas estas interfaces, queda reflejada en la siguiente figura:

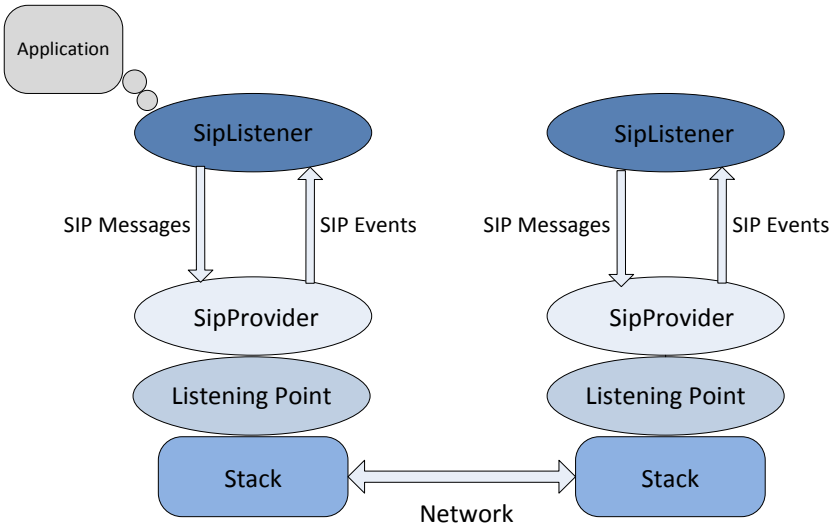

Ilustración 21 Arquitectura de los mensajes Jain SIP (20)

Como se dijo anteriormente, esta arquitectura de mensajes está basada en el modelo asíncrono de eventos *Listener/Provider* (oyente/proveedor). En este modelo, hay una relación directa entre el evento proveedor y el evento usuario, ya que un evento usuario se registra con el evento proveedor.

Los eventos son los encargados de encapsular, tanto peticiones (*Requests*) como respuestas (*Responses*). Además, es un modelo unidireccional, ya que la aplicación no puede enviar eventos de salida (*Requests*), sino mensajes de salida (*Responses*).

Los mensajes de tipo *Request* son aquellos enviados desde el cliente hasta el servidor, mientras que los mensajes de tipo *Response* son los que se envían del servidor al cliente, y en respuesta a un *Request* anterior.

SipListener representa el evento usuario, de forma que se queda a la escucha (entrada de eventos) de mensajes, que pueden pertenecer a un diálogo ya establecido, o pueden referirse a un nuevo diálogo.

SipProvider representa el elemento proveedor que se encarga de recibir los mensajes desde la red y se los envía al nivel de aplicación como eventos.

Además de las interfaces anteriores, JAIN SIP define cuatro factorías utilizadas para la construcción de *Requests* y *Responses* que conforman el protocolo SIP. Estas cuatro factorías son las siguientes (20):

- **SipFactory:** interfaz que define los métodos para crear objetos de la pila y objetos de otras factorías (transacciones y diálogos establecidos dentro de un contexto de SIP).

- **AddressFactory:** interfaz que define los métodos para crear las URIs de SIP (esquema de direcciones necesarias para llamar a otra persona vía SIP).

- **HeaderFactory:** interfaz que define los métodos para crear objetos cabecera (para todas las cabeceras y extensiones de las cabeceras del protocolo SIP).

- **MessageFactory***:* interfaz que define los métodos para crear las peticiones y las respuestas del protocolo SIP.

JAIN SIP está desarrollado de una manera sencilla y flexible, que permite a los programadores la capacidad de modificarlo, o extenderlo, tanto en términos de cabeceras como de métodos, ante nuevas RFCs o *releases* del protocolo SIP.

2.9 *Apache Tomcat*

Apache es un servidor HTTP de código abierto, desarrollado por la ASF (*Apache Software Fundation*) y utilizado como contenedor de servlets y JSPs (*Java Server Pages*). Además, puede ejecutar servicios Web mediante el uso de *Apache Axis* (21).

Las JSPs se caracterizan por estar escritas en código HTML, pero que además pueden contener código fuente escrito en cualquier lenguaje de programación (Java, C…) de manera que son páginas dinámicas que se ejecutan en el servidor en cuestión. Por otro lado, los servlets son únicamente programas Java que se ejecutan en el servidor y cuya misión es recibir peticiones por parte de un cliente y generar una respuesta (orientado a eventos).

2.10 *Lenguajes de Programación*

2.10.1 JavaScript

JavaScript (22) es un lenguaje de programación basado en acciones y orientado a objetos, de manera que es capaz de describir funciones que respondan ante la acción de presionar una tecla, mover el ratón o cargar páginas.

Se utiliza principalmente para crear funciones en el lado del cliente, dentro de una aplicación web (técnicamente, en el lado del cliente, se denomina *Navigator JavaScript*), pero también se utiliza en el lado del servidor y se le denomina *LiveWire JavaScript*.

Aunque JavaScript se desarrolló con una base similar al C, no está relacionado con Java, el cual tiene una semántica completamente diferente, así como propósito.

JavaScript nace a consecuencia de la necesidad de crear un mayor dinamismo en las páginas web, hasta entonces bastante estáticas gracias al protocolo HTML, que no permitían la interactuación entre los usuarios.

El código JavaScript se puede desarrollar dentro de las páginas HTML (o HTML5), dentro de los caracteres *<body></body>*, y acotado por los caracteres *<script></script>*, aunque también puede estar ubicado en un archivo externo, con

extensión “.js” e indicando en la propia página HTML la ubicación de dicho archivo, tal y como se muestra en el siguiente ejemplo (22):

<script type = "text/javascrip" src ="codigoJavaScript.js"></scipt>

La sintaxis es muy similar a la utilizada en C y no necesita de la instalación de ningún *framework*, ya que al estar desarrollado en el lado del cliente, es interpretado por el propio navegador. A continuación se muestra la definición de alguno de los elementos básicos utilizados en el lenguaje (22):

- Variables: *var = “Bienvenido”*
- Condiciones: *if(variable>3){...}*
- Bucles: *for (variable; variable<10; variable++) {...}*

JavaScript es una tecnología fundamental en la web. Junto con la estandarización de la ECMA (*European Computer Manufacturers Association*), y adoptada posteriormente por la ISO (*International Organization for Standarization)* y W3C DOM, JavaScript es considerado un elemento básico para la creación de aplicaciones web dinámicas en el lado del cliente.

2.10.2 HTML y HTML5

HTML (*HyperText Markup Language*) es el lenguaje utilizado en la definición de páginas web. Está formado principalmente por un conjunto de etiquetas entre las cuales se define la estructura y los objetos que conformarán la página.

HTML (23) se basa en hacer referencia a objetos externos, es decir, si queremos incrustar una imagen en nuestra página, no introducimos el código relativo a la imagen en nuestro código HTML, sino que hacemos una referencia de tipo texto a dicho objeto, mediante su ubicación.

En sus inicios, HTML se creó como elemento de divulgación de información, pero no se pensó como un elemento de uso de información multimedia (lo que hoy en día son las *webs*), por lo que las primeras versiones de HTML son bloques más estáticos que los que, por ejemplo, se pueden desarrollar hoy en día con HTML5.

HTML5 (23) es la versión 5 del protocolo HTML, que aún en la actualidad se encuentra en desarrollo. HTML5 establece nuevos objetos que le permiten adaptarse a las nuevas páginas *webs*, mucho más dinámicas y de mayor interactuación con el usuario. HTML5 incorpora etiquetas de tipo *canvas* 2D, 3D, audio y vídeo, con la incorporación de *codecs* para mostrar los datos multimedia, de forma que ofrece mayor capacidad en el manejo y la animación de objetos o imágenes en nuestra página web.

Como conclusión, HTML5 intenta dotar a las páginas *web* actuales, de mucha más versatilidad y flexibilidad, ajustándose a las necesidades demandadas por los usuarios, de forma que se adapta de una manera más eficaz a la gran cantidad de navegadores y plataformas que existen en la actualidad.

2.10.3 XML (Request/Response)

XMLHTTP (*Extensible Markup Language Hypertext Transfer Protocol*) es una interfaz desarrollada para generar peticiones de tipo HTTP y HTTPS (HTTP *Secur*e) a servidores Web (24).

El cliente, puede realizar tantas instancias de la interfaz como necesite, para generar las peticiones suficientes para mantener el diálogo con el servidor. Este diálogo con el servidor se puede realizar de manera síncrona o asíncrona. Si se utiliza de manera síncrona, el programa se detiene hasta que la acción queda completada, sin embargo, si se utiliza de manera asíncrona, el programa no se detiene, sino que se crea un objeto evento, que se queda a la escucha de la acción, hasta que ésta se complete, y mientras tanto, el programa sigue ejecutando otras operaciones.

2.10.4 Java

Java es un lenguaje de programación utilizado para la creación de aplicaciones de una manera sencilla. Estas aplicaciones suelen utilizarse integradas en páginas *web* y ejecutadas en el explorador, de manera que dotan de un mayor dinamismo y funcionalidad a las aplicaciones *web* (25).

Originalmente desarrollado por *Sun Microsystems*, su sintaxis deriva de C y C++, pero sin embargo, es mucho más potente que estos. Las aplicaciones Java son compiladas para su ejecución posterior, independientemente de la arquitectura de la máquina donde se desarrolle dicha compilación. Es extremadamente eficaz, ya que una vez desarrollado el código y compilado, puede ejecutarse desde cualquier dispositivo, de forma que una vez compilado no tiene que volver a compilarse en otra máquina, su funcionalidad ya es completa.

Para el correcto desarrollo de aplicaciones Java, necesitamos disponer de dos elementos:

- JDK (*Java Development Kit*), que es el software que contiene las herramientas necesarias para el desarrollo de las aplicaciones Java, como las variables de entorno (*JAVAPATH* y *CLASSPATH*) y programas de ejecución (*javac.exe, java.exe*, etc.).
- J2SE (*Java 2nd Standard Edition*), que contiene las APIs necesarias para desarrollar nuestra aplicación Java, como por ejemplo, *java.applet* o *java.awt*, entre otras.

2.11 *Servlets*

Los Servlets son objetos que se encargan de generar páginas *web* de manera dinámica según los parámetros que le envíe el navegador. Se almacenan en contenedores de Servlets, que pueden o no ser contenedores web, como por ejemplo, Apache Tomcat (26).

La ejecución de un Servlet se realiza de la siguiente manera:

- Un servidor se inicia y carga el Servlet almacenado en él. Para ello, realiza una llamada al método *init* del Servlet creando una instancia de éste.

- El Servlet comienza a tratar con las peticiones del cliente y genera las respuestas adecuadas para éste.

- Cuando se realiza una petición de cierre del servidor, el Servlet es destruido por éste. Para ello, el servidor utiliza el método *destroy* del propio Servlet.

Todos los Servlets han de implementar a la clase *Servlet* o en su defecto, extender de una clase que implementa dicha interfaz como *HttpServlet*. Dentro de la interfaz *Servlet* se definen los métodos que permiten iniciar el Servlet, configurarlo, atender a las peticiones y generar las respuestas adecuadas y destruir el Servlet.

2.12 *AJAX*

AJAX (*Asynchronous JavaScript And XML*, JavaScript asíncrono y XML) es una técnica, no un lenguaje, utilizada en desarrollo Web para crear páginas web más dinámicas y rápidas (27). Gracias a AJAX, estas páginas web ejecutadas por el cliente en un navegador, pueden ser actualizadas de manera asíncrona mediante un intercambio mucho más reducido de datos con el servidor, es decir, facilita la actualización de una parte de la página web, sin tener que realizar una recarga de la página completa. De esta forma, AJAX mejora tanto la velocidad como la usabilidad de las aplicaciones Web.

Como se ha dicho, AJAX no es un lenguaje, sino que está formado por un conjunto de tecnologías ya existentes (27):

- XMLHttpRequest: utilizado para intercambiar datos de forma asíncrona con el servidor.
- JavaScript/DOM: para mostrar e interactuar de forma dinámica con la información.
- CSS: para el diseño de la información.
- XML: formato de lenguaje usado por lo general, para la transferencia de datos

En AJAX, la interactuación que se lleva a cabo entre el navegador y el servidor se muestra en la Ilustración 22.

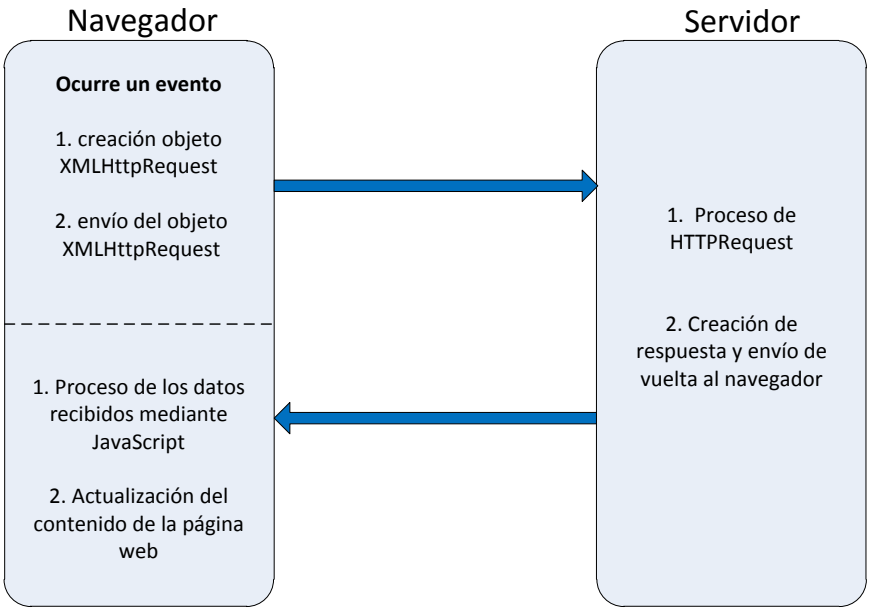

Ilustración 22 Intercambio de datos navegador-servidor AJAX

La parte negativa de AJAX, es que al generar la actualización dinámica de páginas web, estas páginas web no quedan registradas en el historial del navegador o en los registros de navegación, al igual que es posible que ciertos dispositivos, como algunos teléfonos móviles, las páginas generadas con AJAX no se vean de manera correcta.

Parte III: Descripción del trabajo realizado

66

Capítulo 3

Diseño del Sistema VoD

3.1 *Introducción*

En este capítulo del proyecto se describe la estructura del sistema implementado. Se detallan las especificaciones para cada uno de los elementos que lo componen, así como las características principales de cada uno de esos elementos.

El objetivo de este proyecto consiste en el diseño e implementación de un cliente de vídeo bajo demanda compatible con la especificación de TISPAN 183 063 (3). En dicha especificación, se describen todos los procedimientos, entidades y protocolos para proporcionar servicios IPTV basados en una red IMS. Adicionalmente, para poder verificar el correcto funcionamiento del cliente de vídeo bajo demanda, se implementará una versión simplificada de un SCF (*Service Control Function*), que es el servidor SIP que en la especificación de TISPAN y se encarga de ejecutar los procedimientos de señalización con el equipo del usuario, y se desplegará un elemento MF (*Media Function*), que permitirá servir al cliente VoD los vídeos seleccionados por el abonado al servicio.

El diseño de este cliente de vídeo bajo demanda se ha estructurado en dos planos o niveles, siguiendo la filosofía utilizada en la arquitectura de las redes IMS. Se utilizará un plano de control, basado en el protocolo SIP, para autenticar al usuario en la red y solicitar la entrega de alguno de los vídeos disponibles en el catálogo del servicio VoD. Así mismo, el diseño incluye un plano de datos, basado en el modelo de streaming, para realizar la entrega de los vídeos solicitados al cliente VoD.

El cliente VoD será accesible vía Web, lo que facilitará su acceso desde cualquier dispositivo que disponga de un navegador y conexión a Internet. Además, el cliente permitirá solicitar y reproducir el contenido de un vídeo almacenado en un servidor de contenidos, a través de una red con plano de control IMS. Sobre este plano de control se realizará el establecimiento de una sesión multimedia, para llevar a cabo, a continuación, la reproducción del vídeo bajo los parámetros establecidos. Una vez la sesión queda

establecida en el plano de control, la entrega del contenido de vídeo se realiza en el plano de datos utilizando un servicio *streaming*. Este *streaming* se realizará de manera unidireccional, desde el servidor de contenidos al cliente, y será el cliente el que a través de peticiones RTSP controle la reproducción del vídeo, por ejemplo pausándolo o reanudando la reproducción a solicitud del usuario.

En la Ilustración 23 a continuación, se muestran los distintos módulos que componen el cliente desarrollado, así como su integración en la arquitectura IPTV de TISPAN:

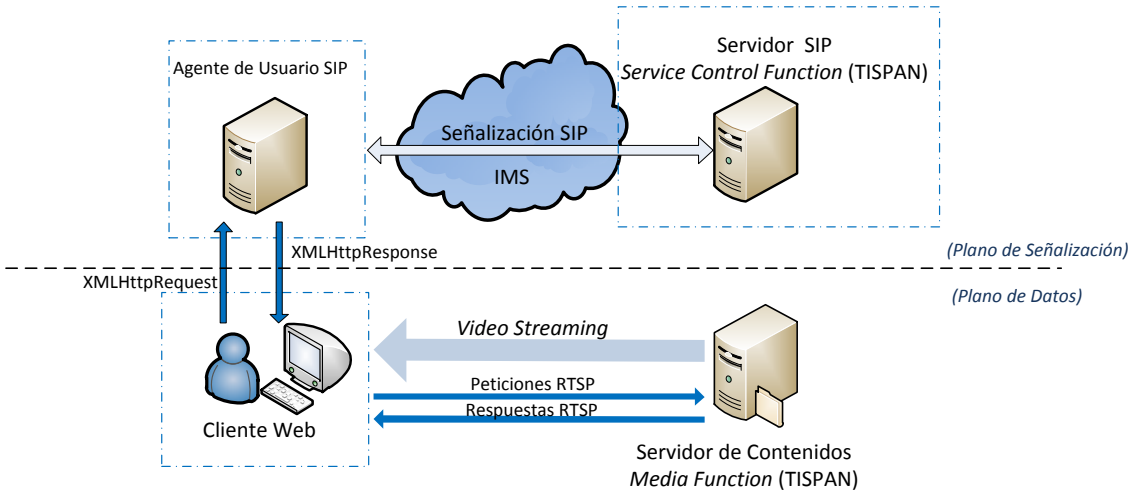

Ilustración 23 Sistema desarrollado para el cliente de vídeo bajo demanda

Como puede observarse, se remarcan en la Ilustración 23 los elementos que han sido desarrollados en este Proyecto, tanto en el lado cliente como en el servidor. En el lado cliente, se incluye un cliente Web y un Agente de Usuario SIP, que hará las veces de Agente de Usuario Cliente (UAC), para iniciar las solicitudes SIP hacia el servidor SCF. En los siguientes apartados se describen las funcionalidades de cada uno de los componentes que integran el sistema de vídeo bajo demanda que se ha desarrollado, haciendo especial hincapié en los módulos correspondientes al diseño del cliente VoD, objeto del presente Proyecto Fin de Carrera.

3.2 *El Cliente Web*

Se describe en este apartado la funcionalidad que implementa el cliente Web. Se diferencia entre los dos planos en los que actúa, el plano de control y el plano de datos, especificando para cada uno de ellos, las funcionalidades llevadas a cabo, así como los requisitos necesarios.

3.2.1 Descripción de funcionalidad en el plano de control

Las funcionalidades que desarrolla el cliente Web en el plano de control, o señalización, son las siguientes:

- Cuando el cliente Web se carga por primera vez en el navegador, contacta con el Agente de Usuarios SIP para activar el proceso de registro del usuario en la red IMS. Este proceso de registro permite autenticar al usuario en la red siguiendo los procedimientos explicados en el apartado 3.3.
- Una vez realizado el registro, el cliente Web ofrece al usuario la posibilidad de elegir diferentes vídeos dentro de un catálogo a través de una interfaz gráfica (ver sección 4.2.1). Tras seleccionar un vídeo, el cliente Web contacta con el Agente de Usuario SIP para iniciar el establecimiento de una sesión multimedia con el servidor SCF (ver sección 3.3). Esta sesión permitirá la entrega del vídeo en el plano de datos.

3.2.2 Descripción de funcionalidad en el plano de datos

Una vez terminados los procesos de registro y establecimiento de la sesión en el plano de control, el usuario podrá visualizar en la interfaz gráfica del cliente Web el contenido demandado.

- Tras el establecimiento de la sesión, el cliente Web ejecutará un diálogo RTSP con el MF. Como resultado de este diálogo, el MF comenzará a trasmitir el contenido del vídeo al cliente Web, que reproducirá dicho contenido a través de la interfaz gráfica proporcionada.
- Una vez el vídeo está siendo visualizado en el cliente Web, éste tendrá las funcionalidades específicas para pararlo, pausarlo o volverlo a reproducir, mediante peticiones del protocolo RTSP, tal y como se define en la especificación de TISPAN.

3.3 *El Agente de Usuario SIP*

Se describe en este apartado la funcionalidad que ofrece el Agente de Usuario SIP. Este agente sólo actúa en el plano de control o señalización. Sus funcionalidades son:

- Realizar el registro del usuario que accede al cliente de Vídeo bajo demanda en la red IMS. Este proceso de registro se realiza mediante los procedimientos de señalización indicados en la Ilustración 24.

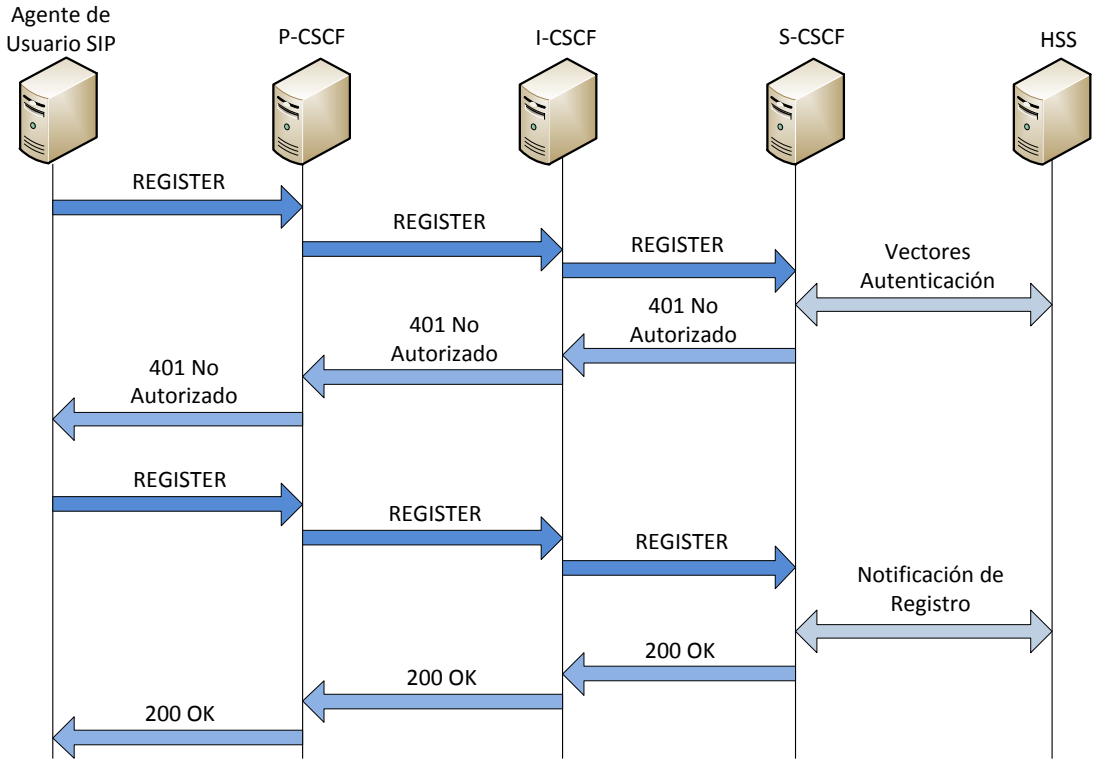

Ilustración 24 Registro del usuario del cliente Web

El proceso de registro se basa en los siguientes pasos de señalización: primero se genera una petición de registro que entra en la red IMS a través de P-CSCF, el I-CSCF y el S-CSCF. Para poder proceder con el registro del usuario, éste debe estar autenticado, para ello el HSS le envía al S-CSCF una serie de parámetros de autenticación que son enviados de vuelta en una respuesta *401 Unauthorized* para que el usuario pueda validarse. Esta respuesta llega al cliente, quien la procesa y vuelve a generar una petición de registro, pero ahora incluyendo la respuesta al desafío de seguridad lanzado por el HSS. Esta petición viaja de nuevo por la red IMS hasta llegar al S-CSCF, quien envía los datos al HSS y éste se encarga se enviarle el perfil de dicho usuario ya registrado en la red. Finalmente, el S-CSCF envía un asentimiento *200 OK* para informar al usuario de que ha quedado validado y registrado.

- Establecimiento de la sesión multimedia mediante el intercambio de peticiones y respuestas del protocolo SIP, a través de la red IMS, con el SCF. Este proceso se realiza mediante los procedimientos indicados en la Ilustración 25

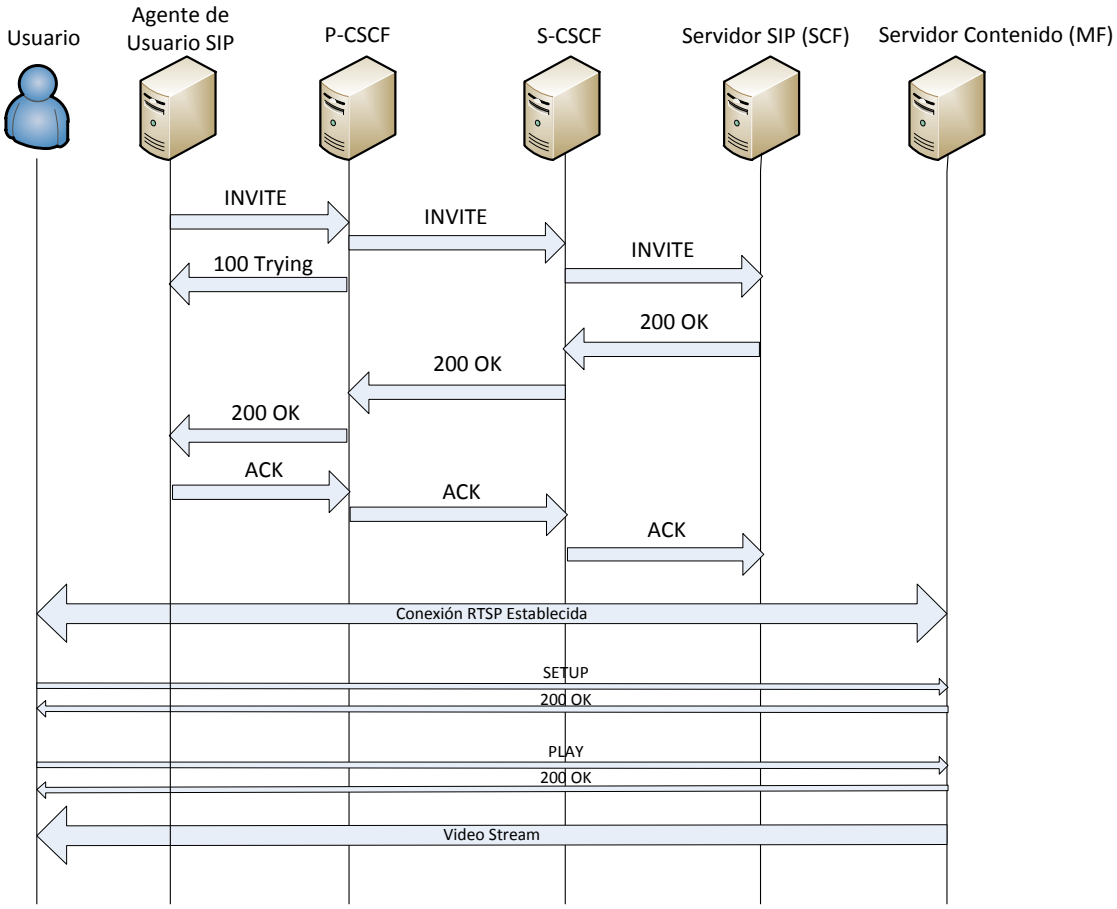

Ilustración 25 Establecimiento de sesión y posterior visualización del vídeo

Una vez se ha realizado el registro del usuario, se procede al intercambio de los mensajes SIP *INVITE*, *OK*, *ACK* para llevar a cabo el establecimiento de la sesión. El Agente de usuario SIP genera una petición *INVITE* que es dirigida a través de IMS al SCF. Una vez que esta petición alcanza al servidor, éste genera una respuesta de tipo *200 OK* que reenvía de nuevo al cliente a través de la red IMS. Cuando la respuesta llega al cliente, éste genera un asentimiento *ACK*, y como consecuencia, se establece la sesion multimedia. Una vez ocurre esto, se envía una petición de RTSP *SETUP*, en la que el cliente Web informa al MF de los parámetros de transporte para la recepción del vídeo según lo intercambiado en la carga SDP del *INVITE* y el *OK*. Finalmente, médiate una petición PLAY, comienza la retransmisión de vídeo desde el servidor de contenido hasta el usuario.

3.4 *El Servidor SCF*

Aunque no es el objetivo de este proyecto, para verificar el correcto funcionamiento del cliente de vídeo bajo demanda, se ha realizado el diseño de un servidor SIP, que atiende a las peticiones que le llegan desde el Agente de Usuario SIP. Este servidor provee un subconjunto de las funcionalidades de un SCF según se recogen en la especificación de TISPAN. En particular, el servidor SCF que se ha desarrollado soporta el intercambio de los mensajes de señalización recogidos en la Ilustración 25.

3.5 *El servidor de contenidos MF*

No es objetivo de este Proyecto el diseño de un servidor de contenidos, por lo que para comprobar la correcta funcionalidad del cliente Web implementado, se utiliza software adicional, explicado en el apartado 4.5, que se encargará de desempeñar la figura del MF. Este servidor de contenidos será el encargado intercambiar peticiones y respuestas RTSP con el cliente Web, así como servir el vídeo elegido, tal y como queda recogido en la Ilustración 25.

Capítulo 4

Implementación de los elementos del Sistema

4.1 *Introducción*

El cliente de Vídeo bajo Demanda (VoD) que se ha desarrollado consta de dos módulos o componentes: un cliente Web y un Agente de Usuario SIP. El cliente Web se encargará de reproducir/controlar la reproducción del vídeo solicitado por el usuario mediante una interfaz gráfica accesible a través de un navegador Web. Este cliente interactúa con un Agente de Usuario SIP, que se ejecuta en el mismo equipo en el que se ejecuta el cliente Web al que accede el usuario, para de este modo iniciar los procesos de registro del usuario en la red IMS y el establecimiento de sesiones multimedia, con el objeto de configurar la entrega del contenido de vídeo desde un servidor de contenidos.

Para poder comprobar el correcto funcionamiento del cliente de vídeo bajo demanda desarrollado, se ha implementado adicionalmente, un servidor SCF, que desempeña la funcionalidad de AS (*Application Server*) en la red IMS y es el encargado de establecer la sesión con el Agente de Usuario SIP. Además, se utilizará un servidor de contenidos MF, que servirá el vídeo al cliente. Hay que puntualizar, que este Proyecto Fin de Carrera que se centra en la implementación del cliente de vídeo bajo demanda, es decir, en la implementación del cliente Web y el Agente de Usuario SIP, no se implementa la comunicación entre el SCF y el MF, simplemente se realiza una implementación básica del SCF para poder verificar el correcto funcionamiento del cliente.

El sistema completo que se ha implementado se muestra en la Ilustración 23 y se estructura en dos planos o niveles:

- Un plano de control, basado en el protocolo SIP, que posibilita el establecimiento de sesiones multimedia a través de la red IMS. Este

plano involucra al cliente Web, al Agente de Usuario SIP y al servidor SCF. Cuando el usuario selecciona un vídeo de entre los disponibles en el catálogo, el cliente Web contacta al Agente de Usuario SIP, de modo que éste inicia el establecimiento de una sesión multimedia con el servidor SCF, utilizando para ello una solicitud INVITE de SIP. Este proceso de señalización se muestra en la Ilustración 26

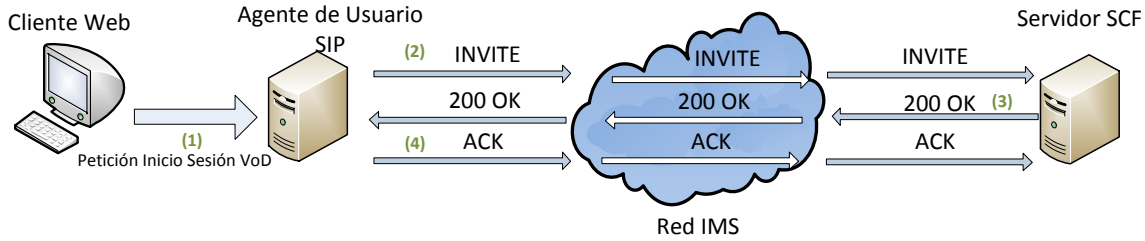

Ilustración 26 Establecimiento de sesión a través de IMS

- Un plano de datos, basado en la transmisión de un flujo multimedia de tipo vídeo, desde el servidor MF hasta el cliente web. La reproducción del contenido de vídeo puede ser controlada por el cliente Web mediante el intercambio de mensajes RTSP (ej. *Play*, *Pause*, *Stop*…) entre éste y el servidor MF. Este plano, representado en la Ilustración 27, no involucra al Agente de Usuario SIP ni al servidor SCF.

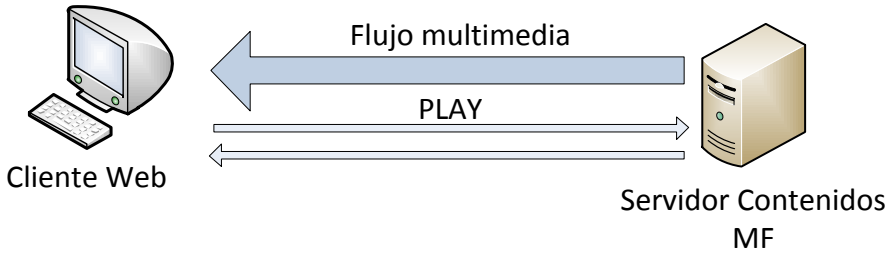

Ilustración 27 Retransmisión de vídeo bajo demanda

A continuación, se describen cada uno de los elementos involucrados en el desarrollo, así como las tecnologías utilizadas para su implementación.

4.2 *El Cliente Web*

El cliente Web está formado por una interfaz gráfica e implementado mediante un Servlet que utiliza el servidor Web Apache Tomcat (ver apartado 2.9) para su funcionamiento. El Servlet se ha desarrollado mediante la implementación de una clase Java (ver apartado 2.10.4) y una clase de tipo XML (ver apartado 2.10.3) que le indica al servidor Apache quién es el Servlet de la aplicación, mientras que la interfaz gráfica del cliente se ha desarrollado mediante el uso de HTML/HTML5 (ver apartado 2.10.2). Para la interacción entre la interfaz gráfica del cliente y el Servlet, se ha desarrollado una clase

Javascript (ver apartado 2.10.1), que utiliza la tecnología AJAX (ver apartado 0) para enviar las peticiones desde la interfaz hasta el Servlet.

En la siguiente tabla (Tabla 2 Conjunto de clases que forman el clientese hace una breve descripción de cada una de estas clases:

Nombre de la clase	Tipo	Descripción
Rtc	Java	Servlet. Recibe las peticiones del cliente Web y las envía al Agente de Usuario SIP
interfazweb	HTML/HTML5	Interfaz gráfica. Establece la página HTML que hace de interfaz para el usuario
main	Javascript	Trata las peticiones realizadas por el usuario sobre la interfaz gráfica e interactúa con el Servlet
web	XML	Tiene una estructura definida usada para configurar el Servlet
DumbVideo	HTML/HTML5	Interfaz de cliente que se obtiene al seleccionar la primera opción de vídeo en interfazweb.html
Google	HTML/HTML5	Interfaz de cliente que se obtiene al seleccionar la segunda opción de vídeo en interfazweb.html
Wild	HTML/HTML5	Interfaz de cliente que se obtiene al seleccionar la tercera opción de vídeo en interfazweb.html

Tabla 2 Conjunto de clases que forman el cliente Web

4.2.1 La interfaz gráfica

La implementación visual del cliente se basa en el desarrollo de una interfaz gráfica utilizando HTML y HTML5 (ver apartado 2.10.2). Cuando el cliente accede a la aplicación, visualiza la interfaz de usuario que se muestra en la Ilustración 28.

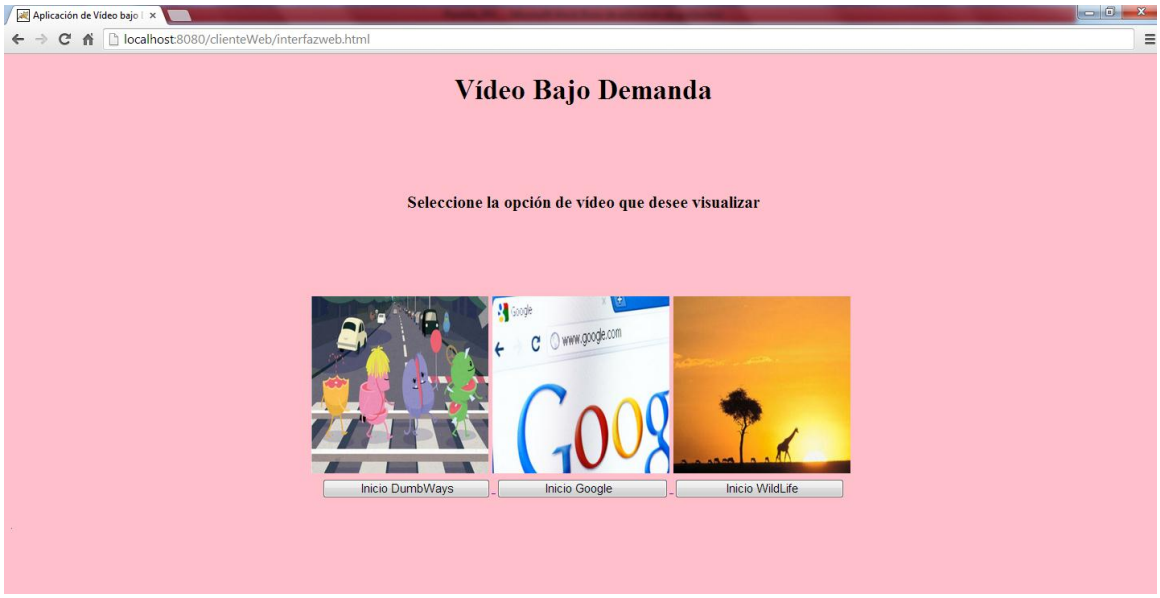

Ilustración 28 Interfaz del cliente 1

En esta primera interfaz, el cliente accede a uno de los contenidos ofrecidos pulsando en el botón que corresponda al vídeo que quiere visualizar. Dichos botones se encuentran en la parte inferior de la interfaz y sobre ellos, una imagen de referencia al vídeo que se visualiza al pulsar sobre cada uno. Una vez el usuario selecciona uno de los vídeos, una nueva interfaz se despliega según se muestra en la Ilustración 29, en la que se visualizará el contenido del vídeo elegido.

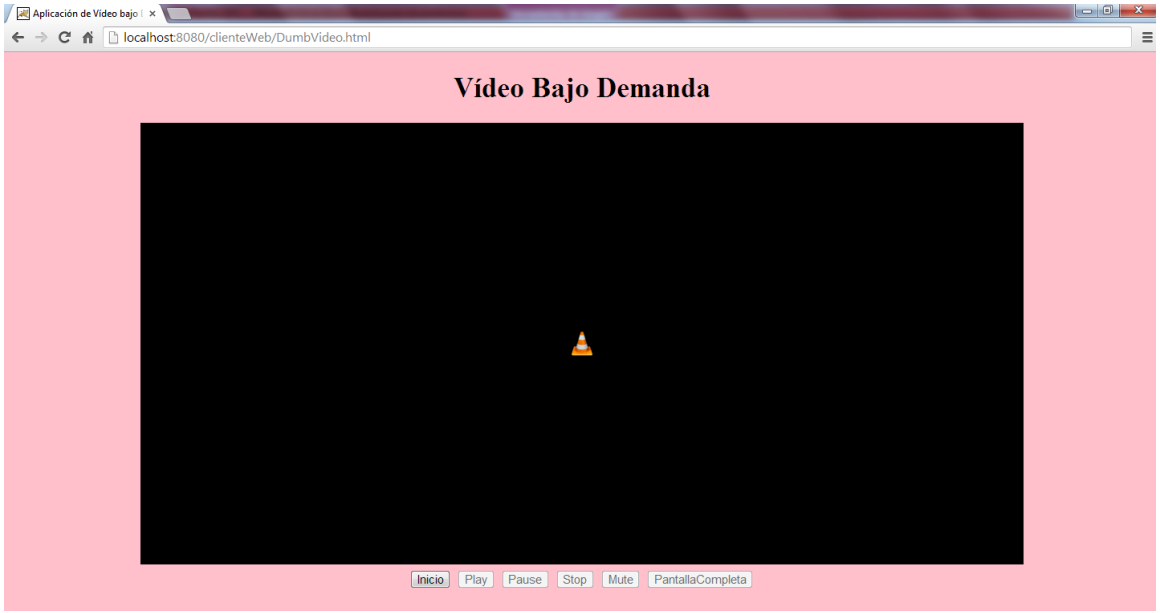

Ilustración 29 Interfaz del cliente 2

Dentro de esta interfaz, tenemos dos partes bien diferenciadas:

- La zona inferior, donde se encuentran todos los botones de acción para llevar a cabo el proceso de inicio de sesión, mediante el botón *Inicio*, y una vez

establecida la sesión, controlar la reproducción del vídeo mediante los botones *Play*, *Stop* y *Pause*.

- Una zona central, donde va embebido el reproductor multimedia VLC y donde se reproducirá el contenido del vídeo.

4.2.2 El Servlet

El Servlet desarrollado para el cliente Web es de tipo HTTP. Recibe peticiones que le llegan desde el cliente Web y emite las respuestas correspondientes mediante el uso de la tecnología AJAX (ver apartado 0). Este tipo de peticiones y respuestas intercambiadas entre el cliente Web y el Servlet son de tipo *HTTPServletRequest* y *HTTPServletResponse*, respectivamente, y para poder hacer uso de ellas, será necesario la utilización de la librería *servlet-api.jar*.

El cliente Web usa un Agente de Usuario SIP que es ejecutado en el mismo equipo que el propio Servlet para iniciar la sesión. Para comenzar con el establecimiento de la sesión el usuario deberá pulsar sobre el botón de *Inicio* que se encuentra en la parte inferior de la aplicación Web (ver Ilustración 29). A continuación se muestra el diagrama de flujo del conjunto de pasos llevados a cabo cuando el cliente pulse sobre el botón de *Inicio*:

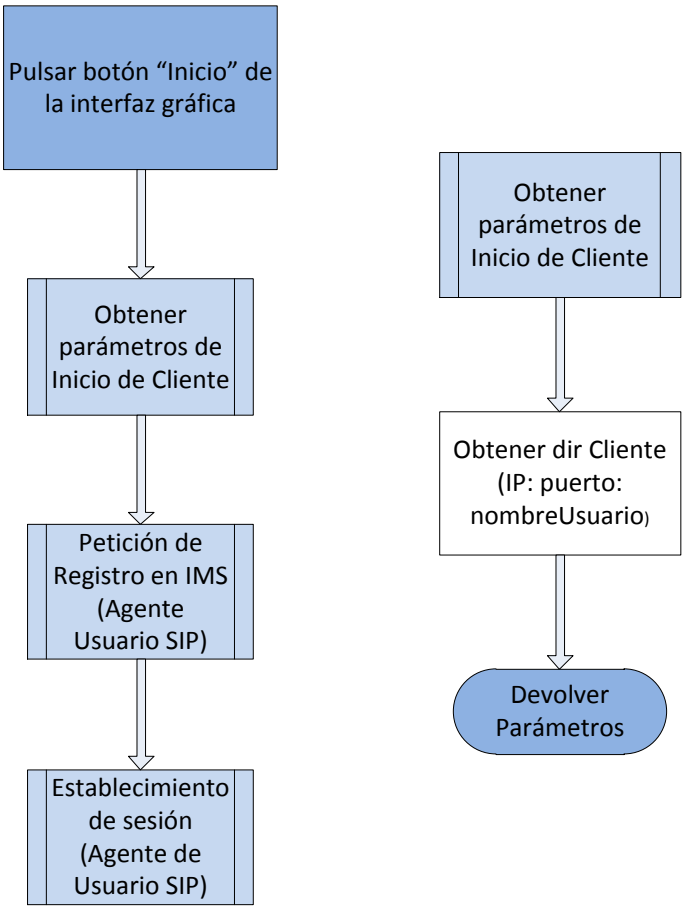

Ilustración 30 Diagrama de flujo de Inicio del cliente

Tras pulsar en el botón de *Inicio*, la orden llegará desde la interfaz gráfica hasta el Servlet. Cuando el Servlet reciba la petición, se encargará de mandarle los parámetros necesarios al Agente de Usuario SIP para comenzar con el registro del usuario en la red IMS y el establecimiento de la sesión con el servidor SCF. En el establecimiento de sesión, la oferta SDP que se genera coincide con la establecida según la especificación de TISPAN (3), apartado 5.1.4, que contiene toda la información necesaria para garantizar que el cliente Web recibe el vídeo y se establece una sesión de control RTSP. Según la especificación, la oferta SDP ha de ser generada como sigue:

- Una línea "*m*" que indica el tipo *stream* RTSP, cuyo formato es:

  m=<media> <port> <transport> <fmt>

  - El campo *media* toma el valor *application*
  - El campo *port* toma el valor por defecto 9, que es el puerto de descarte
  - El campo *transport* debe ser fijado al valor TCP
  - El campo *fmt* debe tomar el valor *iptv_rtsp*

- Una atributo de tipo *a=connection* definido como *new* para establecer que es una nueva conexión
- Un atributo de tipo *a=setup* definido como active para establecer la sesión activa
- Un atributo de tipo *c* que establece el tipo de red, fijado a IN (INERNET), el tipo de dirección IP, fijado a IP4 (IPv4) y la dirección IP relativa al flujo desde el que se envía el contenido de control RTSP (en este caso, la IP del Cliente Web).
- Una línea m que indica el medio de tipo vídeo que se quiere recibir, su formato es:

  m=<media> <port> <transport>

  - El campo *media* toma el valor video
  - El campo *port* toma el valor de inicio del puerto de recepción
  - El campo *transport* toma el valor RTP/AVP

Finalmente, tras establecer la sesión, se establece la conexión TCP para RTSP y el cliente Web y el servidor de contenidos MF intercambian las peticiones *SETUP* y *PLAY*. A partir del momento en el que se envía el PLAY, el cliente puede comenzar a controlar la reproducción mediante los comandos de RTSP.

4.3 *El Agente de Usuario SIP*

El Agente de Usuario SIP es ejecutado en el mismo equipo que el Servlet y recibe de él las peticiones que le llegan del cliente a través de la interfaz gráfica. Está formado por un conjunto de clases Java mostradas en la Tabla 3 que son utilizadas para el intercambio de los mensajes SIP necesarios para el registro y el establecimiento de sesión:

Nombre clase	Tipo	Descripción
SipLayer	Java	Utilizada para la Generación de las

		peticiones INVITE/ACK a intercambiar con el servidor SCF para el establecimiento de la sesión
Register	Java	Realiza el Registro del usuario en la red IMS

Tabla 3 Conjunto de clases que forman el Agente de Usuario SIP

Para implementar la señalización SIP, se ha hecho uso de API de Java JAIN SIP (ver apartado 0). Gracias a esta API podemos generar e intercambiar mensajes de SIP de una manera sencilla y abstrayéndonos de la complejidad que reside en la sintaxis de dichos mensajes (19) y (20). Para ello se utilizará un *SIP Provider*, que recibirá las peticiones, las procesará transformándolas en eventos y entregará dichos eventos a la aplicación, y un *SIP Listener* que gestionará esos eventos para que la aplicación pueda interpretarlos.

La librería utilizada por el Agente de Usuario SIP para el registro del cliente en la red IMS, así como las librerías de la API de Jain SIP que hemos utilizado para implementar el Agente de Usuario SIP y el servidor SCF, son las siguientes:

- *commons-codec-20041127.091804.jar:* utilizada para el registro del usuario en la red IMS. Esta librería fue desarrollada por Francisco Javier de Pablos Gómez, durante su Proyecto Fin de Carrera (28).

- *jainSipApi1.2.jar* y *jainSipRi1.2.jar*: clases principales, interfaces de SIP y referencias de implementación.

- *concurrent.jar*: para la gestión de procesos concurrentes de la aplicación.

- *log4j-1.2.8.jar*: para la gestión de los *logs* de *debug* y los avisos de la aplicación.

A continuación, se describe tanto el registro como el establecimiento de sesión llevado a cabo por el Agente de Usuario.

4.3.1 El Registro

El primer paso a realizar por el Agente de Usuario SIP tras la llegada de la petición por parte del Servlet, es el registro del usuario en la red IMS. Este proceso se muestra en la Ilustración 31. La implementación del proceso de registro en la red IMS no forma parte del desarrollo de este Proyecto, pero sí se indican los pasos seguidos para el entendimiento del mismo.

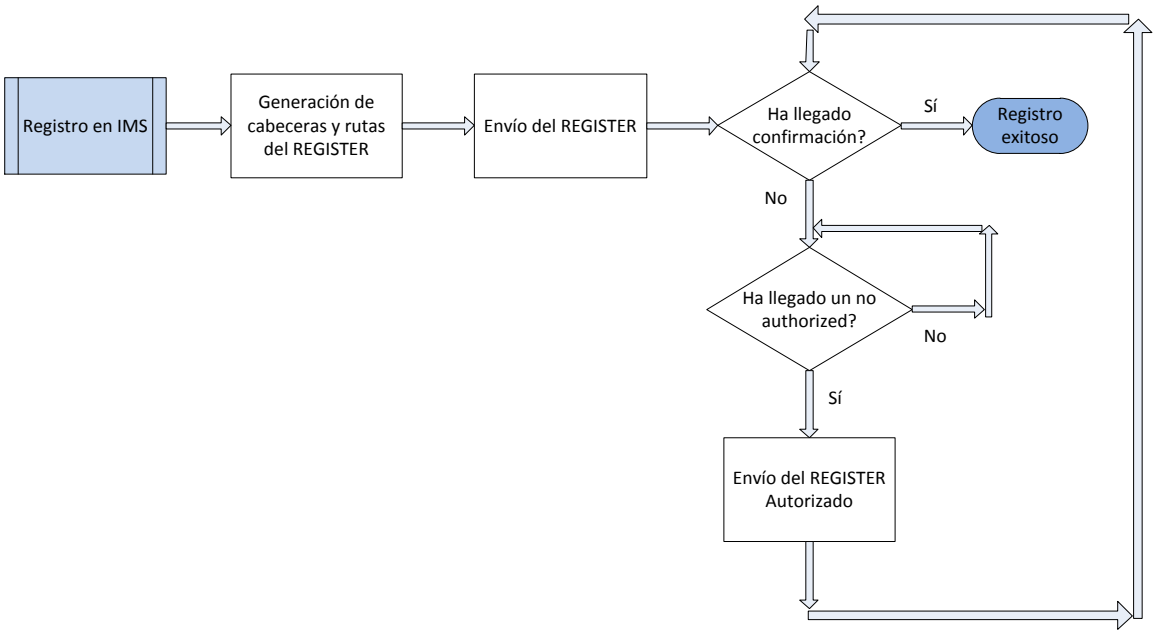

Ilustración 31 Diagrama de flujo del registro del cliente en IMS

En el proceso de registro en la red IMS, tal y como se explicó en el apartado 3.3, el Agente de Usuario genera una primera petición de registro (solicitud *REGISTER* de SIP) que envía a la red IMS. El S-CSCF en la propia red IMS será el encargado de autorizar o no el registro, dependiendo de si el usuario puede ser autenticado. Si el usuario no puede ser autenticado, el S-CSCF genera una respuesta de tipo *Unathorized* que contendrá los parámetros de autenticación necesarios para que el Agente de Usuario vuelva a emitir otra solicitud de registro correcta, para la cual sí recibirá una respuesta confirmando que el usuario ha quedado registrado.

4.3.2 Establecimiento de sesión

Una vez se ha concluido el registro en la red IMS por parte del Agente de Usuario, éste puede iniciar el establecimiento de la sesión multimedia necesaria para garantizar la correcta recepción del vídeo en el cliente Web. Para ello, el Agente de Usuario SIP genera y envía una solicitud *INVITE* a través de la red IMS hasta el servidor SCF. Esta solicitud INVITE incluirá la oferta SDP desarrollada tal y como se ha descrito en el apartado 4.2.2. Observamos el proceso de envío del *INVITE* en la Ilustración 32.

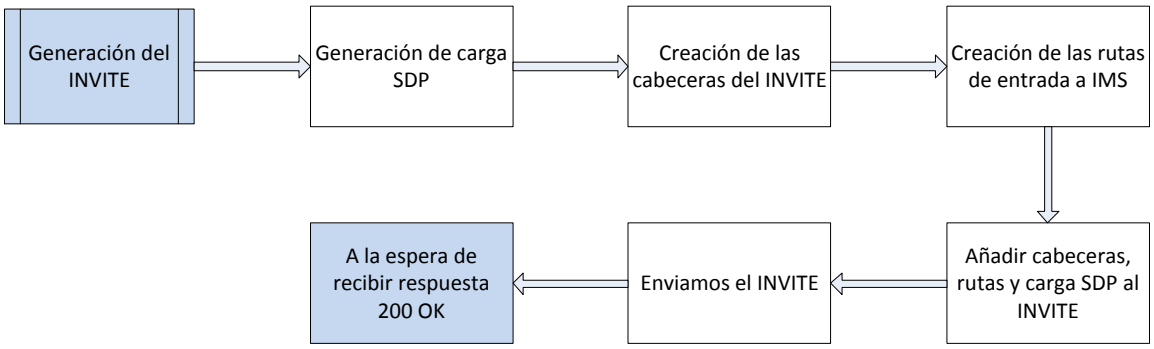

Ilustración 32 Diagrama de flujo de la generación del INVITE por parte del Agente de Usuario

Cuando el Agente de Usuario recibe una respuesta *200 OK* por parte del servidor SCF, procesa dicha respuesta y genera una solicitud de confirmación *ACK*:

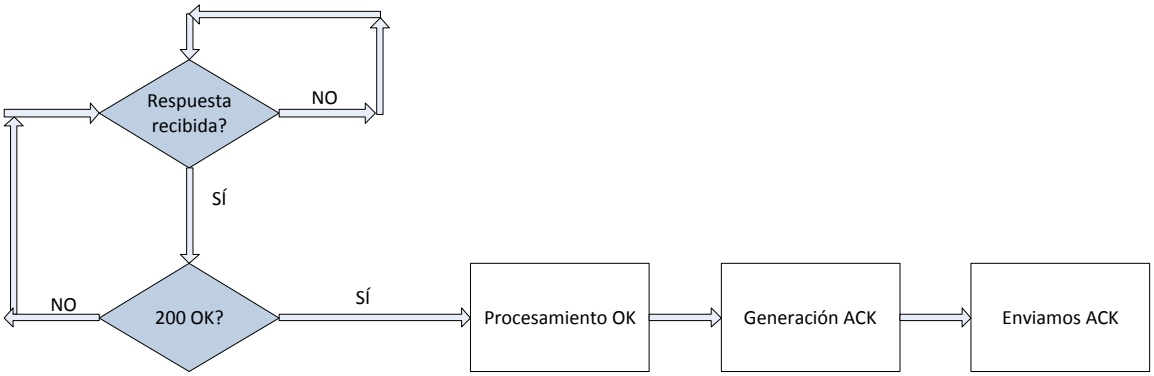

Ilustración 33 Procesamiento de respuesta y envío de ACK

Una vez la sesión queda establecida, el cliente está listo para comenzar a recibir el vídeo emitido por el servidor de contenidos. El vídeo llega al cliente en la dirección IP y el puerto especificados en la carga SDP del *INVITE* enviado por el Agente de Usuario SIP.

4.4 *El Servidor SCF*

Para comprobar la funcionalidad de la aplicación desarrollada, se ha implementado la funcionalidad básica de un servidor SCF, mediante un conjunto de clases Java que se muestran en la Tabla 4 y que dotan al servidor de capacidad suficiente para procesar las solicitudes de SIP que llegan desde el Agente de Usuario y generar la correspondiente respuesta. En este caso, no se ha desarrollado una interfaz gráfica, ya que no es necesario para esta parte del proyecto.

Nombre clase	Tipo	Descripción
Rtc	Java	Es el método principal de la aplicación del servidor SIP. Inicializa los datos del servidor SCF (IP, puerto y

		nombre de AS)
SipLayer	Java	Espera a recibir peticiones tipo *INVITE* por parte del Agente de Usuario y las procesa generando a continuación la respuesta *200 OK*

Tabla 4 Conjunto de clases que forman el Servidor SCF

Al igual que ocurre para el caso del Agente Usuario SIP, para poder implementar la señalización SIP, se ha hecho uso de la interfaz JAIN SIP (ver apartado 0), implementándose las siguientes librerías:

- *jainSipApi1.2.jar* y *jainSipRi1.2.jar*: clases principales, interfaces de SIP y referencias de implementación.
- *concurrent.jar*: para la gestión de procesos concurrentes de la aplicación.
- *log4j-1.2.8.jar*: para la gestión de los *logs* de *debug* y los avisos de la aplicación.

A continuación, se describe el proceso de recepción de la petición *INVITE* y la consecuente generación de la respuesta *200 OK* por parte del servidor SCF.

4.4.1 Establecimiento de sesión: generación de *200 OK*

Inicialmente, el servidor SCF es lanzado e inicializado, quedándose a la espera de recibir una petición de tipo INVITE por parte del Agente de Usuario. Este proceso de inicio del servidor SCF queda reflejado en la Ilustración 34 mostrada a continuación:

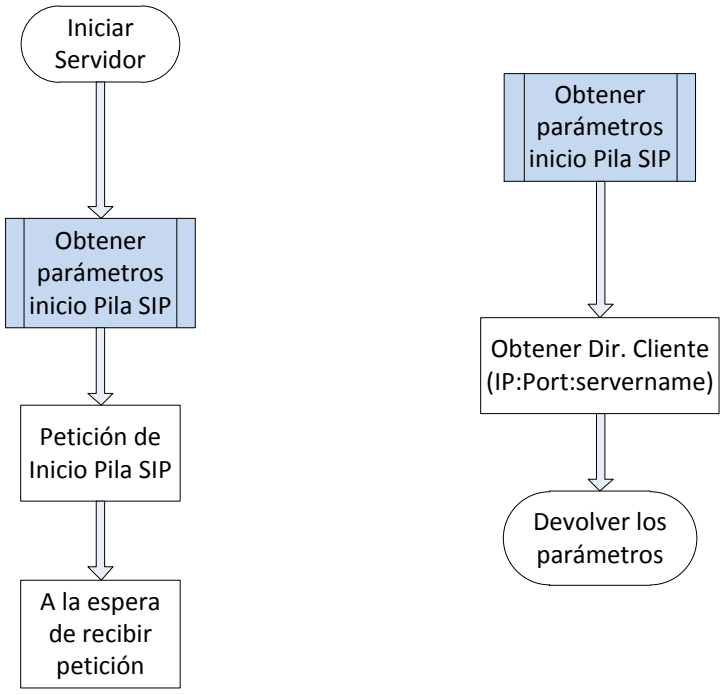

Ilustración 34 Diagrama de flujo de inicio del servidor SCF

Una vez que las variables están inicializadas, se inicializa la pila SIP del servidor y éste se queda a la espera de recibir una solicitud *INVITE*. Una vez haya recibido el *INVITE* por parte del cliente y generado la respuesta *200 OK*, el servidor SCF permanece a la espera de la recepción de una solicitud *ACK* por parte del Agente de Usuario SIP, tal y como se muestra en la Ilustración 35.

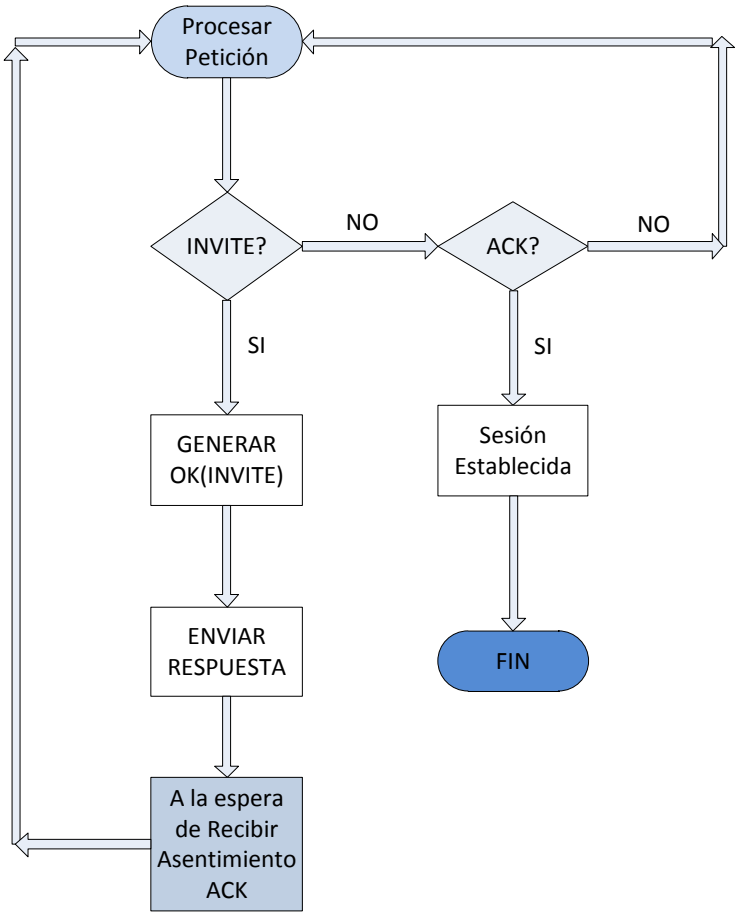

Ilustración 35 Diagrama de flujo del proceso del servidor SCF

Cuando llega un *INVITE*, el servidor analiza la carga SDP que contiene ese *INVITE* de donde obtiene el puerto y la IP a través de los cuales el cliente quiere recibir el vídeo desde el servidor de contenidos. En ese momento, el servidor SCF generará su respuesta *200 OK* con la carga SDP correspondiente, aceptando la oferta SDP que ha llegado en el *INVITE* y la envía al Agente de Usuario SIP.

El siguiente mensaje que debe recibir el servidor tras el envío del *200 OK*, es un *ACK* por parte del cliente, confirmando que queda establecida la sesión.

4.5 *El servidor de Contenidos MF*

Aunque no es objeto de este proyecto la implementación de un servidor de contenidos MF, se describen en este apartado la aplicación software adicional utilizada

para representarlo, así como una breve explicación del funcionamiento del mismo, para facilitar el entendimiento al lector.

El servidor de contenidos se implementa mediante el uso de la aplicación VLC en modo servidor. No se ha desarrollado ninguna aplicación Java o interfaz para este apartado, ya que no ha sido necesario, ni es objetivo de este proyecto. La configuración necesaria para utilizar la aplicación VLC como servidor se describe en el apartado 0.

Una vez se ha realizado el registro y el establecimiento de la sesión, el cliente está preparado para recibir el vídeo que será enviado por el servidor de contenidos. A su vez, el servidor de contenidos recibirá las peticiones de tipo RTSP que le lleguen desde el cliente para proceder al envío o parada del vídeo. El proceso del servidor de contenidos se muestra en la Ilustración 36.

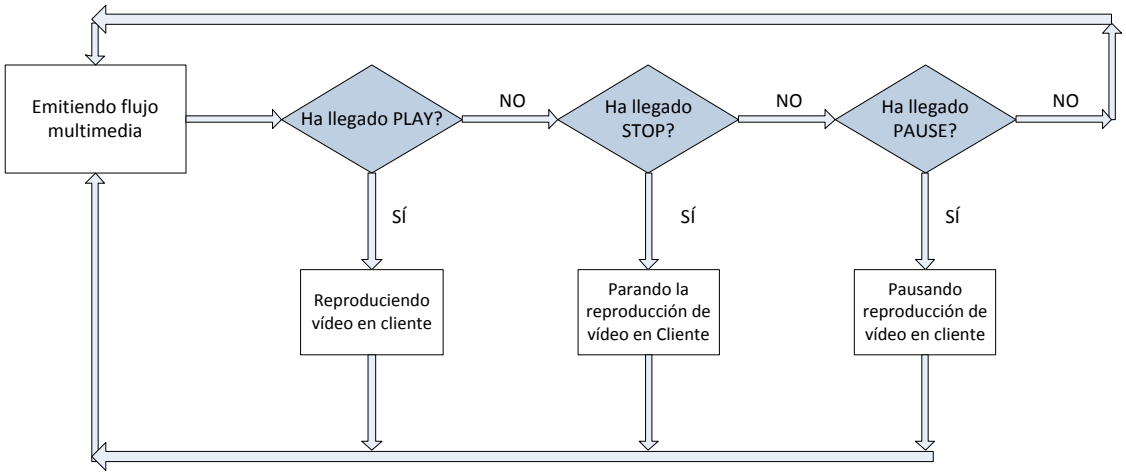

Ilustración 36 Diagrama de flujo del envío de vídeo por parte del servidor de contenidos

Para la reproducción del vídeo en el cliente Web, se utiliza un elemento embebido de tipo VLC y el protocolo RTSP implementado sobre dicho elemento para realizar las acciones *PLAY*, *PAUSE* y *STOP* sobre el flujo de datos. Para poder utilizar el reproductor VLC con capacidad RTSP no ha sido necesario el uso de ninguna librería adicional, ya que por su característica de poder ser "embebido" en HTML5, en la propia interfaz de cliente Web, es suficiente para que se puedan enviar las peticiones RTSP desde el cliente Web hasta el servidor de contenidos.

Capítulo 5

Escenario de pruebas

En este capítulo se describe el escenario de pruebas que se ha utilizado para demostrar el correcto funcionamiento del sistema implementado, y en particular del cliente de Vídeo bajo Demanda objeto del presente proyecto Fin de Carrera. Como ya se ha comentado, los mensajes de señalización SIP se intercambian a través de una red IMS, entre el Agente de Usuario SIP y el servidor SCF. De este modo, en el escenario de pruebas utilizado se ha incluido un Core IMS real (29), que se ha desplegado en un equipo del laboratorio de Proyectos Fin de Carrera del Departamento de Ingeniería Telemática de la Universidad Carlos III de Madrid. El Core IMS utilizado provee la implementación de los elementos de control de llamadas de IMS (es decir, el P-CSCF, el S-CSCF y el I-CSCF), así como un HSS. El escenario de pruebas desplegado se muestra en la Ilustración 37.

Se detallan en la Tabla 5 los parámetros de configuración relevantes de cada uno de los elementos que intervienen en las pruebas.

Elemento	Dir. IP	Puerto SIP	Puerto Vídeo
Cliente Web	163.117.140.34	---	1766 (RTP)
Agente de Usuario	163.117.140.34	5060	---
Servidor SCF	163.117.140.220	5060	---
Servidor Contenidos	163.117.140.39	---	5554 (RTSP)
P-CSCF	163.117.166.11	4060	---
S-CSCF	163.117.166.11	6060	---

Tabla 5 Especificación de los elementos que intervienen en las pruebas

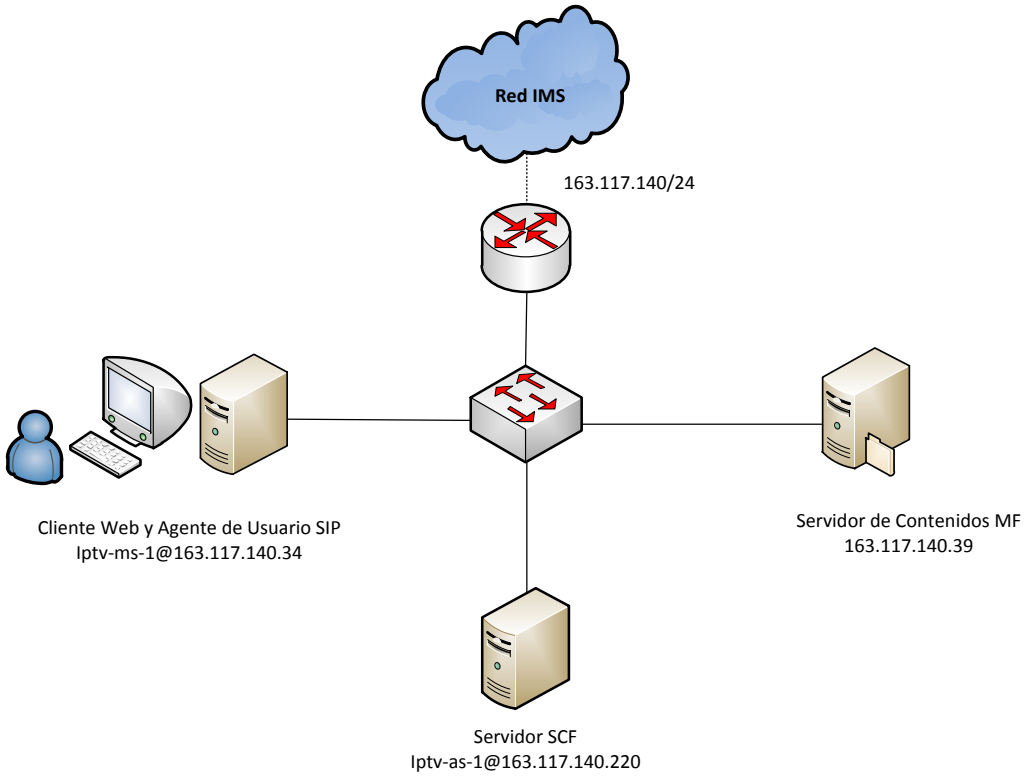

Ilustración 37 Entorno de pruebas

Durante las pruebas, y para facilitar el entorno, el cliente Web y el Agente de Usuario SIP se ejecutaron en el mismo equipo, aunque en la realidad, se ejecutarían en equipos diferentes.

Con respecto al cliente, éste se divide en dos partes: el cliente Web y el Agente de Usuario SIP (éste último se ejecuta en el mismo equipo que el servidor Web Apache). El usuario de la aplicación, tiene un identificador de usuario (una URI SIP pública) al igual que el resto de los elementos del sistema, de la forma:

sip:iptv-ms-1@ims.net

El Agente de Usuario SIP va a ser el encargado de realizar el registro del usuario en la red IMS, y de establecer la sesión SIP en su nombre, para que éste pueda recibir el vídeo en el cliente Web. Para establecer dicha sesión el Agente de Usuario SIP generará un INVITE que se mandará a una URI SIP del tipo:

sip:user@domain

Donde *user* identifica el contenido que se quiere visualizar y *domain* especifica el nombre de dominio del proveedor de servicio VoD.

Por su parte, el P-CSCF y el I-CSCF tienen también cada uno una URI SIP, configuradas en el core IMS. Una vez que se mande el INVITE desde el Agente de Usuario SIP hacia el servidor SCF, éste se redigirá a través de la red IMS por el P-CSCF, que a su vez se lo enviará al S-CSCF, el cual, a través de los filtros establecidos dentro de

la red IMS, los filtros iFC (*Initial Filter Criteria*), lo enrutará hasta el SCF. Cabe destacar que estos filtros están almacenados en el HSS, como parte del perfil del usuario, y que se han definido para las pruebas.

A continuación, se describen cada una de las pruebas que se han realizado para validar la funcionalidad de la aplicación de vídeo bajo demanda basada en la especificación de TISPAN.

5.1 *Registro del usuario en la red IMS*

Antes de proceder al establecimiento de la sesión, el usuario que accede al cliente Web debe registrarse en la red IMS. Este registro es realizado por el Agente de Usuario SIP en nombre del usuario, cuando el usuario pulsa el botón de "*Inicio*" en la interfaz gráfica proporcionada por el cliente web (ver Ilustración 29).

Tal y como se comentó en el apartado 3.3, el registro está formado por cuatro pasos, mostrados en la Ilustración 24. A continuación se muestran las tramas capturadas que corresponden al registro del cliente, así como el contenido de cada una de ellas:

- Petición *REGISTER*:

```
REGISTER sip:ims.net SIP/2.0
Call-ID: reg///1-11351@163.117.140.34
CSeq: 1 REGISTER
From: "iptv-ms-1" <sip:iptv-ms-1@ims.net>;tag=1
To: "iptv-ms-1" <sip:iptv-ms-1@ims.net>
Via: SIP/2.0/UDP 163.117.140.34:5060;branch=z9hG4bK252789b499378bd38fb672ddf8786ab0
Max-Forwards: 20
Contact: "iptv-ms-1" <sip:iptv-ms-1@163.117.140.34:5060>;expires=3600
Authorization: sip:iptv-ms-1@ims.net
Supported: path
User-Agent: SIPp 3.1
Route: <sip:163.117.166.11:4060;lr>
Content-Length: 0
```

En esta primera trama, el Agente de Usuario genera una petición SIP de tipo *REGISTER*, que según se observa en el campo de cabecera *Route* se encamina al P-CSCF, que se encuentra dentro de la red IMS. Una vez envía la petición, se queda a la espera de recibir respuesta.

- Respuesta *Unauthorized*:

```
SIP/2.0 401 Unauthorized - Challenging the UE
Call-ID: reg///1-11351@163.117.140.34
CSeq: 1 REGISTER
From: "iptv-ms-1" <sip:iptv-ms-1@ims.net>;tag=1
To: "iptv-ms-1" <sip:iptv-ms-1@ims.net>;tag=a7612f6ffb1f1cb7233c130a1b198462-3694
Via: SIP/2.0/UDP
163.117.140.34:5060;rport=5060;branch=z9hG4bK252789b499378bd38fb672ddf8786ab0
Path: <sip:term@pcscf1.ims.net:4060;lr>
Service-Route: <sip:orig@scscf1.ims.net:6060;lr>
Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REFER,SUBSCRIBE,NOTIFY,PUBLISH,MESSAGE,INFO
Server: Sip EXpress router (2.1.0-dev1 OpenIMSCore (i386/linux))
```

```
Warning: 392 163.117.166.11:6060 "Noisy feedback tells:  pid=8381 req_src_ip=163.117.166.11
req_src_port=5060 in_uri=sip:scscf1.ims.net:6060 out_uri=sip:scscf1.ims.net:6060 via_cnt==3"
WWW-Authenticate: Digest qop="auth,auth int",
nonce="b8db6df8d69bc6108dbc124917e5d118",realm="ims.net",algorithm=MD5
Content-Length: 0
```

Una vez que los elementos de la red IMS (P-CSCF y el S-CSCF) reciben la petición de registro por parte del usuario, y el S-CSCF pide las autorizaciones necesarias al HSS, se genera una respuesta con un mensaje *Unauthorized* (no autorizado) que contiene la información para llevar a cabo un desafío de seguridad al usuario. A partir de la información incluida en esta trama, el usuario deberá reintentar el registro respondiendo al reto de seguridad proporcionado.

- Petición *REGISTER* con autorización:

```
REGISTER sip:ims.net SIP/2.0
Call-ID: reg///1-11351@163.117.140.34
CSeq: 2 REGISTER
From: "iptv-ms-1" <sip:iptv-ms-1@ims.net>;tag=1
To: "iptv-ms-1" <sip:iptv-ms-1@ims.net>
Via: SIP/2.0/UDP 163.117.140.34:5060;branch=z9hG4bK81630aaec45162aa12aa01003f8f9ce1
Max-Forwards: 20
Contact: <sip:163.117.140.34:5060>;expires=3600
Authorization: Digest
response="cf9c0e584b8cc3f1bec977ae0fada3ce",cnonce="6b8b4567",username="iptv-ms-
1@ims.net",nc=00000001,qop=auth,nonce="b8db6df8d69bc6108dbc124917e5d118",realm="ims.net",uri
="sip:163.117.166.11:4060",algorithm=MD5
Route: <sip:163.117.166.11:4060;lr>,<sip:orig@scscf1.ims.net:6060;lr>
Supported: path
User-Agent: SIPp 3.1
Content-Length: 0
```

El Agente de usuario inicia un nuevo registro respondiendo al desafío de seguridad, para autenticar al usuario ante la red IMS, de modo que se genera una nueva solicitud *REGISTER* que se envía hacia el P-CSCF.

- Respuesta *200 OK:*

```
SIP/2.0 200 OK - SAR succesful and registrar saved
Call-ID: reg///1-11351@163.117.140.34
CSeq: 2 REGISTER
From: "iptv-ms-1" <sip:iptv-ms-1@ims.net>;tag=1
To: "iptv-ms-1" <sip:iptv-ms-1@ims.net>;tag=a7612f6ffb1f1cb7233c130a1b198462-077f
Via: SIP/2.0/UDP
163.117.140.34:5060;rport=5060;branch=z9hG4bK81630aaec45162aa12aa01003f8f9ce1
P-Associated-URI: <sip:iptv-ms-1@ims.net>
Contact: <sip:163.117.140.34:5060>;expires=3600
Path: <sip:term@pcscf1.ims.net:4060;lr>
Service-Route: <sip:orig@scscf1.ims.net:6060;lr>
Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REFER,SUBSCRIBE,NOTIFY,PUBLISH,MESSAGE,INFO
Server: Sip EXpress router (2.1.0-dev1 OpenIMSCore (i386/linux))
Warning: 392 163.117.166.11:6060 "Noisy feedback tells:  pid=8379 req_src_ip=163.117.166.11
req_src_port=5060 in_uri=sip:scscf1.ims.net:6060 out_uri=sip:scscf1.ims.net:6060 via_cnt==3"
Content-Length: 0
```

Tras recibir la respuesta de SIP *200 OK* a la solicitud *REGISTER*, el usuario ha quedado registrado en la red IMS, concluyendo por tanto satisfactoriamente el proceso de registro.

5.2 *Establecimiento de la sesión multimedia*

Una vez el cliente ha quedado registrado en la red IMS, el Agente de Usuario pasa a realizar el intercambio de mensajes de señalización SIP con el SCF, como resultado del cual se establecerá una sesión multimedia entre el servidor MF y el cliente Web.

Este proceso de señalización es iniciado por el Agente de Usuario SIP, mediante el envío de una solicitud *INVITE*, incluyendo una oferta SDP, al servidor SCF a través de la red IMS:

```
INVITE sip:iptv-as-1@ims.net SIP/2.0
Call-ID: e874ef171a3b5b3de470f5a16f26fd5c@163.117.140.34
CSeq: 1 INVITE
From: <sip:iptv-ms-1@ims.net>;tag=sqmtsp1nrdhc7bhsqtuv
To: <sip:iptv-as-1@ims.net>
Via: SIP/2.0/UDP 163.117.140.34:5060
Max-Forwards: 70
Content-Type: application/sdp
Route: <sip:orig@163.117.166.11:4060;lr>, <sip:orig@scscf1.ims.net:6060;lr>
Content-Length: 782

v= 0
o=cliente 1 IN IP4 163.117.140.34
m=application 9 TCP iptv_rtsp
c= IN IP4 163.117.140.34
a=setup:active
a=connection:new
m= video 1766 RTP/AVP 31
c= IN IP4 163.117.140.34
a=recvonly
```

Analizando el *INVITE*, podemos observar lo siguiente:

- Se redirige a través de la red IMS por el P-CSCF, cuya dirección es 163.117.166.11, que será el encargado, dentro de la red, de reenviárselo al S-CSCF y éste a su vez, reencaminarlo hasta el servidor SCF gracias a los filtros establecidos dentro de la red IMS, los filtros iFC. Concretamente, estos filtros están almacenados en el HSS como parte del perfil del usuario y son consultados por el S-CSCF para enviar correctamente el *INVITE* a su destinatario, el servidor de aplicación *iptv-as-1*.

- Tal y como se indica en la cabecera *Content-Type*, el cuerpo del *INVITE* está formado por la carga SDP que será la encargada de establecer los parámetros de la sesión IPTV según la especificación de TISPAN.

- La carga SDP nos indica el tipo de sesión multimedia que se desea establecer por parte del cliente Web. Esta carga SDP está formada según lo establecido en el apartado 5.1.4 de la especificación 183 063 de TISPAN (3). Se establece una conexión TCP para RTSP sobre una dirección IP del cliente, donde se recibirá el

vídeo emitido por el servidor de contenidos. Se establece también el canal de vídeo, indicando el protocolo de transporte (RTP), el número de puerto, inicialmente 0 y el formato de codificación empleado AVP 31 por ser vídeo.

Una vez recibido el *INVITE* por parte del Servidor SCF, éste lo procesa, examinando los parámetros de la carga SDP y generando una respuesta *200 OK* indicando en la oferta SDP los siguientes parámetros, según la especificación de TISPAN:

```
SIP/2.0 200 OK
Record-Route: <sip:mo@scscf1.ims.net:6060;lr>,<sip:mo@pcscf1.ims.net:4060;lr>
Call-ID: 2070da836661663c340cb364686b19ea@163.117.140.34
CSeq: 1 INVITE
From: <sip:iptv-ms-1@163.117.140.34>;tag=sqmtsp1nrdhc7bhsqtuv
To: <sip:iptv-as-1@163.117.140.220>
Via: SIP/2.0/UDP 163.117.166.11:6060;branch=z9hG4bK4e5b.fd529086.0,SIP/2.0/UDP
163.117.166.11:4060;branch=z9hG4bK4e5b.8e2ebe07.0,SIP/2.0/UDP
163.117.140.34:5060;rport=5060;branch=z9hG4bKc088d525fe3f43fedbe219e528ec4010
Content-Type: application/sdp
Content-Length: 782

v= 0
o=cliente 1 IN IP4 163.117.140.34
m=application 5554 TCP iptv_rtsp
c= IN IP4 163.117.140.220
a=fmtp:rtsp: rtsp://163.117.140.39:5554/stream
a=setup:passive
a=connection:new
m= video 1766 RTP/AVP 31
c= IN IP4 163.117.140.220
a=sendonly
```

- Las líneas *m* y *c* han de ser idénticas a las recibidas en la oferta SDP que llegó en el INVITE desde el Agente de Usuario.
- Se añade una línea a=fmtp:rtsp que indica la dirección absoluta del servidor MF.
- Se debe incluir un atributo de tipo *a=sendonly*
- Debe incluir los mismos atributos de tipo *a* que incluía la oferta original del *INVITE* (uno de ellos con el valor para la conexión de tipo *passive*).
- El Servidor SCF añade una línea *a* de tipo *control* para indicar al cliente Web el servidor de contenidos remoto.

Finalmente, el Agente de Usuario SIP recibe la respuesta *OK* por parte del servidor SCF, y confirma el establecimiento de la sesión mediante el envío de una solicitud *ACK:*

```
ACK sip:iptv-as-1@163.117.140.220 SIP/2.0
Via: SIP/2.0/UDP 163.117.140.34:5060;branch=z9hG4bKbcaeba4479b1122718c3fab1e86700f6
CSeq: 1 ACK
Call-ID: e874ef171a3b5b3de470f5a16f26fd5c@163.117.140.34
From: <sip:iptv-ms-1@163.117.140.34>;tag=sqmtsp1nrdhc7bhsqtuv
To: <sip:iptv-as-1@163.117.140.220>;tag=1d079b52
Max-Forwards: 70
Route: <sip:mo@pcscf1.ims.net:4060;lr>,<sip:mo@scscf1.ims.net:6060;lr>
Content-Length: 0
```

A partir de este momento, la sesión queda establecida y el cliente Web puede comenzar a recibir el vídeo desde el servidor de contenidos. Para ello, no tendrá más que pulsar el botón *Play* tal y como se muestra en la Ilustración 38.

5.3 *Control RTSP de la reproducción del vídeo*

Cuando la sesión ha quedado establecida, los botones de control de vídeo que aparecían inactivos en la pantalla de inicio de la interfaz del cliente (ver Ilustración 29), se activan, para que el cliente Web pueda a partir de ese momento, comenzar a recibir el flujo de vídeo y controlar su reproducción.

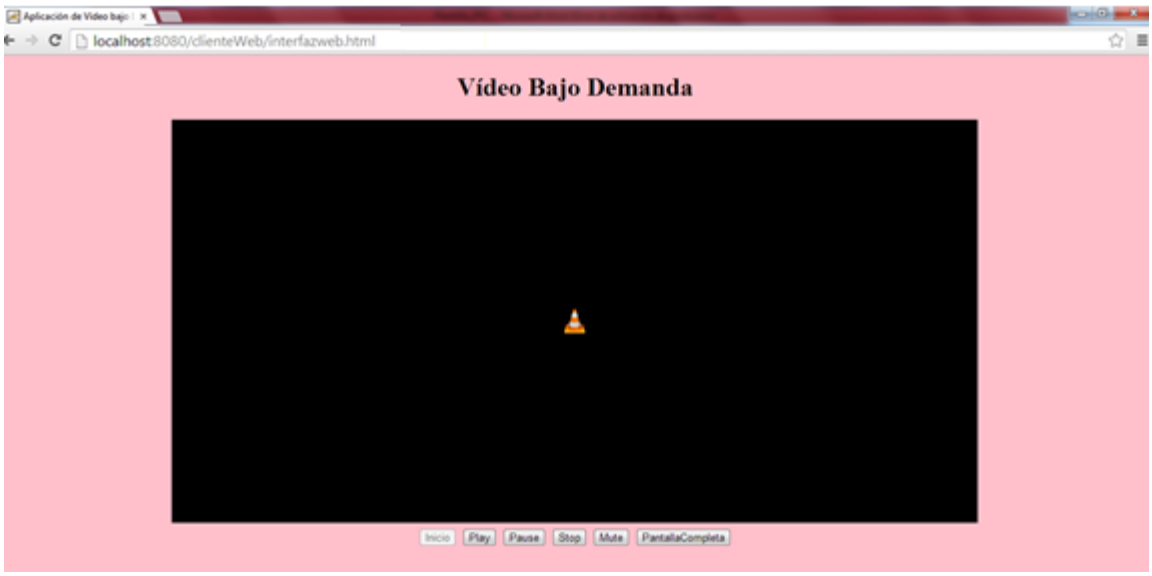

Ilustración 38 Interfaz del cliente lista para recibir flujo de vídeo

La primera vez que el usuario pulsa el botón Play, se genera la primera petición de tipo RTSP, que coincide, en este caso, con una petición de tipo *SETUP* al reproductor de contenidos. A continuación del SETUP, se enviará una petición de tipo *PLAY*.

En la petición *SETUP*, el cliente envía una trama donde especifica el vídeo que quiere reproducir y del que espera recibir una descripción por parte del servidor. Por su parte, el servidor contesta a esta petición mediante una descripción específica del recurso solicitado. En esta descripción, el servidor indica los flujos multimedia necesarios para la reproducción. A continuación, se muestra un ejemplo de una trama de tipo *SETUP* capturada durante las pruebas. La descripción que se establece coincide con la establecida por la oferta SDP que se intercambiaron el Agente de Usuario y el servidor SCF:

En esta petición, se especifica cómo va a ser transportado el flujo de datos multimedia, que también ha quedado establecido por la oferta SDP intercambiada entre el Agente de Usuario SIP y el servidor SCF:

No.	Time	Source	Destination	Protocol	Length	Info
87	17.0380020	163.117.140.34	163.117.140.39	RTSP	239	SETUP rtsp://163.117.140.39:5554/stream/trackID=4 RTSP/1.0
89	17.1623720	163.117.140.39	163.117.140.34	RTSP	320	Reply: RTSP/1.0 200 OK

Ilustración 39 Captura Wireshark SETUP RTSP

La petición del cliente:

```
Request: SETUP  rtsp://163.117.140.39/stream RTSP/1.0
CSeq: 4
User-Agent: LibVLC/2.0.8 (LIVE555 Streaming Media v2012.12.18)
Transport: RTP/AVP; unicast; client_port=1766-1767
```

La respuesta del servidor de contenidos:

```
Response: RTSP/1.0 200 OK
Server: VLC/2.1.0
Date: wed, 02 Oct 2013 16:32:48 GMT
Transport: RTP/AVP;unicast;client_port=1766-1767; server_port=1374-
1375;ssrc=90A47738;mode=play
Session:dfe5d493f3c348a2
Content-length:0
Cache-Control: no-cache
CSeq: 4
```

Como se puede observar en las tramas, se establecen los puertos para el cliente y el servidor para la recepción de datos de tipo RTP, así como la especificación de transporte. El cliente añade en el *SETUP* la dirección *rtsp://163.117.140.39* recibida en la respuesta SDP que le llegó en el 200 OK por parte del servidor SCF.

Tras el intercambio de la petición *SETUP* se genera una petición de tipo *PLAY*. Esta petición desencadenará que el cliente comience a recibir el flujo de vídeo enviado desde el servidor y comenzará su reproducción. En las pruebas, se realizó una de las siguientes capturas ante una petición tras pulsar el botón Play:

No.	Time	Source	Destination	Protocol	Length	Info
96	17.3703950	163.117.140.34	163.117.140.39	RTSP	224	PLAY rtsp://163.117.140.39:5554/stream RTSP/1.0
122	17.6218970	163.117.140.39	163.117.140.34	RTSP	408	Reply: RTSP/1.0 200 OK

Ilustración 40 Captura Wireshark PLAY RTSP

La petición por parte del cliente:

```
Request: PLAY rtsp://163.117.140.39:5554/stream RTSP/1.0
Method: PLAY
URL: rtsp://163.117.140.39:5554/stream
CSeq: 6
User-Agent: LibVLC/2.0.8 (LIVE555 Streaming Media v2012.12.18)
Session: dfe5d493f3c348a2
```

La respuesta por parte del servidor:

```
Response: RTSP/1.0 200 OK
Statur: 200
Server: VLC/2.1.0
Date: wed, 02 Oct 2013 16:32:48 GMT
RTP-Info: url-rtsp://163.117.140.39/stream/trackID=4; seq=53139;rtptime=96987000,
url=rtsp://163.117140.39:5554/stream/reackID=5;seq=48986;rtptime=473823630
Session: dfe5d493f3c348a2
Content-Length: 0
```

Una vez estamos reproduciendo el video en el cliente, este flujo de datos puede ser parado mediante una petición de tipo *TEARDOWN*. Tras esta petición, se produce una parada de la reproducción del vídeo. Una captura tomada durante las pruebas tras realizar un *Stop* del vídeo se muestra a continuación:

No.	Time	Source	Destination	Protocol	Length	Info
4360	43.8840710	163.117.140.34	163.117.140.39	RTSP	209	TEARDOWN rtsp://163.117.140.39:5554/stream RTSP/1.0
4367	43.9056790	163.117.140.39	163.117.140.34	RTSP	220	Reply: RTSP/1.0 200 OK

Ilustración 41 Captura Wireshark TEARDOWN RTSP

La petición del cliente:

Request: TEARDOWN rtsp://163.117.140.39:5554/stream RTSP/1.0
Method: TEARDOWN
URL: rtsp://163.117.140.39:5554/stream
CSeq: 8
User-Agent: LibVLC/2.0.8 (Live555 Streaming Media v2012.12.18)
Session: dfe5d493f3c348a2

La respuesta del servidor:

Response: RTSP/1.0 200 OK
Status: 200
Server: VLC/2.1.0
Date: wed, 02 Oct 2013 16:33:15 GMT
Session: dfe5d493f3c348a2
CSeq: 8

Tras el intercambio de estas tramas, el cliente Web deja de recibir el contenido enviado desde el servidor, por lo que si desea volver a recibir dicho contenido, deberá pulsar de nuevo sobre el botón Play, lo que desencadenará un nuevo intercambio de tramas SETUP y PLAY.

Finalmente, se adjunta en la Ilustración 42, una captura de la reproducción del vídeo, en la interfaz que ve el cliente.

Ilustración 42 Reproducción del vídeo bajo demanda

94

Parte IV: Conclusiones

96

Capítulo 6

Conclusiones y líneas futuras

En este capítulo se recogen las prinicipales conclusiones obtenidas al término de este Proyecto Fin de Carrera, así como un conjunto de líneas de trabajo futuras.

6.1 *Conclusiones*

Este Proyecto Fin de Carrera ha consistido en el desarrollo de un cliente de vídeo bajo demanda compatible con la especificaciones de TISPAN 182 027 (2) y TISPAN 183 063 (3), para sistemas IPTV sobre redes IMS. Este cliente VoD se ha desarrollado para que sea accesible vía Web, de forma que se facilite su uso desde cualquier dispositivo que disponga de un navegador y conexión a Internet. El diseño del cliente VoD se ha estructurado en dos planos, atendiendo a la filosofía utilizada en la arquitectura de las redes IMS, un plano de control y un plano de datos. El plano de control está basado en el protocolo SIP, utilizado para el registro del usuario en la red y el establecimiento de sesiones de vídeo bajo demanda. El plano de datos está basado en el modelo de streaming para la entrega del vídeo solicitado al cliente VoD. El desarrollo del cliente de vídeo bajo demanda se ha dividido en dos partes:

- El cliente Web, que interactúa con un Agente de Usuario SIP para proceder al registro del usuario en la red IMS y ofrecerle, a través de una interfaz gráfica, la posibilidad de elegir un vídeo dentro de un catálogo establecido. Una vez el usuario selecciona el contenido que quiere visualizar, el cliente Web contacta de nuevo con el Agente de Usuario SIP que establecerá una sesión multimedia con un servidor SCF. Tras el establecimiento de la sesión, el cliente Web establecerá un diálogo RTSP con un servidor de contenidos MF, de manera que éste comenzará a transmitirle el contenido del vídeo, que será reproducido a través de la interfaz gráfica del cliente Web. Una vez la visualización del vídeo haya comenzado, el cliente Web facilitará las funcionalidades específicas al usuario para pausarlo, pararlo o volverlo a reproducir, mediante peticiones del protocolo RTSP.

- El Agente de Usuario SIP, que es el encargado de realizar el registro, en la red IMS, del usuario que accede al cliente de vídeo bajo demanda, así como llevar a cabo el establecimiento de la sesión multimedia mediante el intercambio de peticiones y respuestas del protocolo SIP, con el servidor SCF, a través de la red IMS.

Adicionalmente, para poder verificar el correcto funcionamiento del cliente de vídeo bajo demanda, se ha implementado la funcionalidad básica del plano de control de un servidor SCF, según la especificación de TISPAN. Se ha desplegado también un servidor de contenidos (MF), basado en la herramienta VLC.

Destacamos también en las conclusiones que en el enfoque inicial de este Proyecto Fin de Carrera se consideró el uso de una nueva tecnología para la implementación del mismo, la tecnología WebRTC. La idea principal era utilizar los fundamentos de WebRTC para la gestión de descripciones de sesión multimedia, como para la visualización del vídeo transmitido desde el servidor de contenidos. Al tratarse de una tecnología novedosa, tras analizar la funcionalidad de esta tecnología se descartó su uso para los fines de implementación del proyecto, ya que las APIs disponibles principalmente estaban orientadas a implementar de manera sencilla aplicaciones de audio y vídeo interactivo (ej. videollamada), resultando complejo su uso para el tipo de servicio objeto de este trabajo, Vídeo bajo Demanda.

6.2 *Líneas futuras*

A partir del desarrollo que se ha realizado del cliente de vídeo bajo demanda, se pueden establecer una serie de líneas futuras de trabajo, para introducir nuevas fucnionalidades o mejoras en el sistema desarrollado. Algunas de estas posibles líneas futuras de trabajo se describen a continuación:

- Implementación y desarrollo del resto de elementos que conforman la arquitectura de TISPAN como son el servidor SCF y el servidor de contenidos MF. Como se ha descrito a lo largo de la memoria, se ha implementado de una manera básica y sencilla un servidor SCF y se ha utilizado software adicional para representar la función del servidor MF. Es por esto, que una posible línea de implementación futura consistiría en la implementación y desarrollo completos de un servidor SCF y un servidor MF aptos para la interactuación con el cliente de vídeo bajo demanda.

- Desarrollo de las funcionalidades de control correspoondientes a la terminación de sesión. Los procedimientos de plano de control presentados en este Proyecto Fin de Carrera, se corresponden con el registro y establecimiento de ssesiones multimedia. Para disponer de una implementación completa del plano de control, sería necesario sin embargo desarrollar los procedimientos relacionados con la terminación de la sesión.

- Implementación de otros servicios complementarios sobre el sistema desarrollado de cliente de vídeo bajo demanda, definidos en la especificación de TISPAN 182 027 (2), como son por ejemplo el desarrollo de un servicio de presencia de

usuario, servicios de perfiles de usuario y personalización, notificaciones, canales personalizados, etc.

- Implementación de un cliente de vídeo bajo demanda mediante el uso de la tecnología WebRTC. Como se ha dicho con anterioridad, se descartó el uso de esta tecnología al principio del proyecto, sin ebargo, dado el potencial de WebRTC y su creciente popularidad, podría resultar interesante proporcionar un cliente Web basado en esta tecnología.

Parte V: Anexos

Anexos

Anexo A: Presupuesto

A continuación, se lista el conjunto de tareas necesarias para el desarrollo de este proyecto, especificando la duración de cada una de ellas, así como el número de recursos necesarios utilizados.

- **Tarea 1: Deficinión de objetivos y requisitos.**

  Esta tarea se basa en la definición de objetivos y requisitos, es decir, sabemos qué queremos desarrollar, una aplicación de vídeo bajo demanda basada en la especificacioes 182 027 y 183 063 de TISPAN para sistemas IPTV, y estudiamos cómo la queremos desarrollar, necesitamos establecer los protocolos y herramientas que vamos a utilizar.

  - Plano de control y señalización: protocolos SIP, SDP, IMS y herramientas JAIN SIP, la red IMS de la Universidad Carlos III de Madrid y el conjunto de lenguajes de programación necesarios programar al cliente (Java, Javascript, HTML, XML).
    Duración: 3 semanas

  - Plano de datos: protocolos RTP y RTSP y herramientas VLC en modo servidor.
    Duración: 2 semanas

- **Tarea 2: Desarrollo del Estado del arte.**

  Una vez acabada la tarea anterior, en la que se ha desarrollado un esquema general de los protocolos y herramientas que se creen necesarios para el desarrollo del sistema, se comienza el estudio más detallado de cada uno de esos protocolos y herramientas que se utilizarán.

  - Estudio de SIP/SDP: peticiones y respuestas para el establecimiento de sesión genérico entre cliente y servidor.
    Duración: 1 semana

  - Estudio de RTSP: relación de SDP con RTSP para generar la oferta adecuada al intercambio de vídeo RTSP que se produce en el plano

de datos y aplicación a IPTV, según la especificación de TISPAN 183 063 (3).
Duración: 2 semanas

- IMS: estudio de la integración de SIP/SDP/RTSP dentro de la red IMS.
  Duración: 2 semanas

- VoD: estudio de las diferentes opciones de realizar la implementación de VoD siguiendo las especificaciones de TISPAN para la estructura del sistema IPTV.
  Duración: 2 semanas

- Lenguajes de programación: conocimiento de los lenguajes de programación utilizados, así como la utilización de un Servlet y de la tecnología AJAX para la implementación del cliente.
  Duración: 2 semanas

- **Tarea 3. Diseño del sistema VoD.**

  Una vez tenemos el estudio general de lo que se va a desarrollar y conocemos de manera exhaustiva la forma de utilizar los protocolos y las herramientas que nos ayudarán a implementarlo, pasamos al proceso de especificación detallada de los requisitos del sistema. Es decir, llega el momento de tomar las decisiones de diseño.

  - Estudio de los requerimientos del cliente Web
    Duración: 1,5 semana

  - Estudio de los requerimientos Agente de Usuario SIP
    Duración: 1,5 semana

  - Estudio de los requerimientos del servidor SCF y el servidor MF
    Duración: 1 semana

- **Tarea 4: Implementación del sistema**

  Una vez definido el diseño del sistema, pasamos al proceso de implementación.

  - El cliente Web: desarrollo de la interfaz gráfica del cliente y el Servlet que lo implementa.
    Duración: 3 semanas

  - El Agente de Usuario SIP: desarrollo de la señalización para el registro y el establecimiento de la sesión, mediante el protocolo SIP/SDP a través de la red IMS.
    Duración: 3 semanas

- Desarrollo del servidor SCF: establecimiento de la señalización para llevar a cabo el establecimiento de sesión con el Agente de Usuario SIP.
  Duración: 2 semanas

- Servidor de contenidos MF: no hay implementación relativa a este apartado, ya que se utiliza el software de VLC (30) para la implementación
  Duración: 0 semanas

- **Tarea 5: Pruebas**

  Una vez ha sido desarrollado el sistema completo formado por cliente Web, Agente de Usuario SIP, servidor SCF y el servidor de contenido MF, pasamos a realizar las pruebas pertinentes que den la validación a la implementación.

  - Pruebas de validación a nivel de señalización y control: registro del cliente en la red IMS y establecimiento de la sesión llevada a cabo entre Agente de Usuario y servidor SCF
    Duración: 2 semanas

  - Pruebas de validación a nivel de datos, recepción correcta de flujo de datos de vídeo entre cliente Web y el servidor de contenidos, y validación del control de dicho flujo de datos mediante RTSP.
    Duración: 2 semanas

Tarea	Duración	Recursos
Estudio del entorno de desarrollo	5 semanas	1 ingeniero/4 horas diarias
Estudio del estado del arte de las tecnologías implicadas	9 semanas	1 ingeniero/4 horas diarias
Identificación de los requisitos	4 semanas	1 ingeniero/4 horas diarias
Implementación del Sistema	8 semanas	1 ingeniero/4 horas diarias
Pruebas	4 semanas	1 ingeniero/4 horas diarias
Total	30 semanas	1 ingeniero/600 horas

Tabla 6 Planificación de tareas para el desarrollo del proyecto

Se obtienen un total de 30 semanas de trabajo. Considerándose las semanas de 5 días laborables y 4 horas de trabajo diarias, un ingeniero necesitará 600 horas en total, para el desarrollo completo del sistema (cerca de 8 meses).

No se tienen en cuenta costes materiales para calcular el coste total del proyecto, ya que se supone que lo que se necesita es el desarrollo a nivel aplicativo, por lo que sólo se tiene en cuenta el coste por ingeniero. Bajo este supuesto, atendiendo a los valores descritos en el informe del COIT (Colegio Oficial de Ingenieros de Telecomunicación) (31) podemos establecer que el salario medio mensual de un Ingeniero Superior de Telecomunicaciones, con menos de 5 años de experiencia es de unos 2083,33 euros. Por tanto, por 8 meses de trabajo, el presupuesto final es de 16666,64 euros:

	Bruto mensual (€)	Total meses	Total (€)
1 Ingeniero	2083.33	8	16666,64

Tabla 7 Presupuesto del desarrollo

Anexo B: Manual de usuario

En este anexo se explica cómo ejecutar la aplicación de vídeo bajo demanda. Tenemos cuatro elementos diferenciados: el cliente Web, el Agente de Usuario SIP, el servidor SCF y el servidor de contenido.

Antes de poder lanzar la aplicación, necesitaremos disponer de una versión de Java adecuada para la compilación y ejecución de la misma (versiones de Java superiores a la 1.6 son suficientes). Necesitaremos también disponer de las librerías mencionadas en los capítulos 4.2, 4.3 y 4.4, es decir, *jainSipApi1.2.jar*, *jainSipRi1.2.jar*, *concurrent.jar*, *log4j-1.2.8.jar*, *commons-codec-20041127.091804.jar* y *servlet-api.jar*.

Necesitamos además, en el lado del cliente Web, que se disponga del servidor Apache Tomcat instalado en la máquina desde donde se ejecute (se recomienda versiones superiores a Apache 6.0). Es importante saber, que para que la aplicación funcione, es necesario que las librerías nombradas anteriormente (exceptuando la librería para el uso de Servlets, que ya va incluida en Apache), se incluyan dentro de la carpeta /lib del servidor Apache.

- Compilación y ejecución del cliente: nuestro cliente está formado por un cliente Web y un Agente de Usuario SIP. Ambos corren en el mismo equipo o sistema, por lo que nuestro proyecto va a estar formado por varias clases Java, una clase Javascript, clases HTML y un archivo XML (ver Tabla 2), por lo que vamos a realizar la compilación y la ejecución mediante el uso de la herramienta Eclipse.

  Dentro de Eclipse, ha de establecerse la siguiente relación de carpetas de un proyecto tipo *Dynamic Web Proyect*:

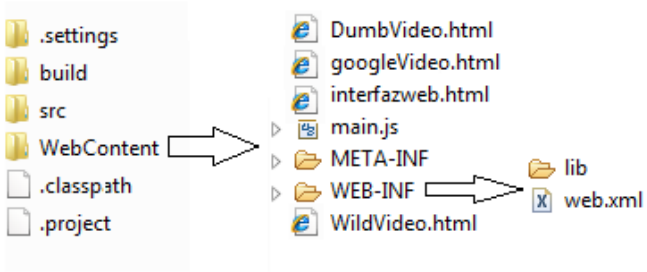

Ilustración 43 Proyecto cliente Web en Eclipse

Dentro de la carpeta src se encuentra el paquete rtc, que almacena las clases Java. En WebContent se almacenan los archivos HTML y Javascript. Finalmente, en

WebContent/WEB-INF se encuentra el fichero .XML. Importamos las librerías a la carpeta de nuestro proyecto en Eclipse tal y como se muestra a continuación:

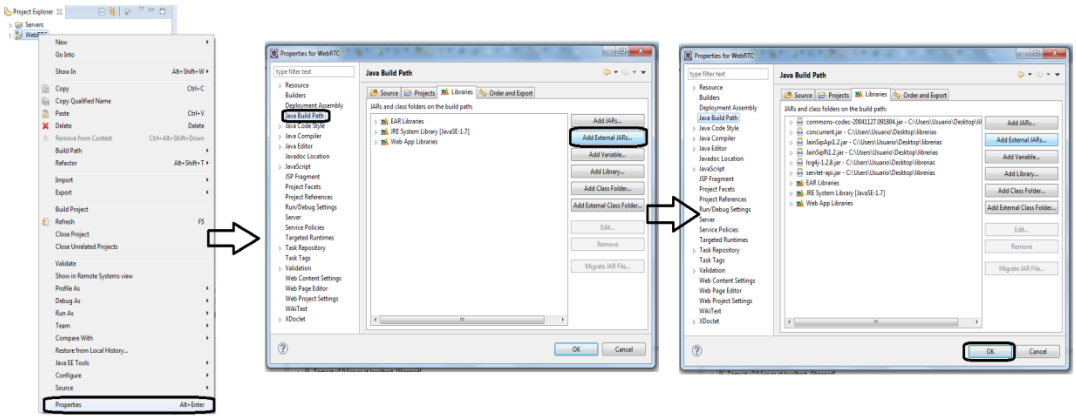

Ilustración 44 Añadir librerías a Eclipse

Sobre el proyecto, botón derecho, *Properties*. En la nueva ventana, *Java Build Path/Add External Jars*, seleccionamos las librerías que queremos importar, aceptamos, y finalmente pulsamos *OK*.

Añadimos el servidor Apache Tomcat al proyecto Web realizando los siguientes pasos:

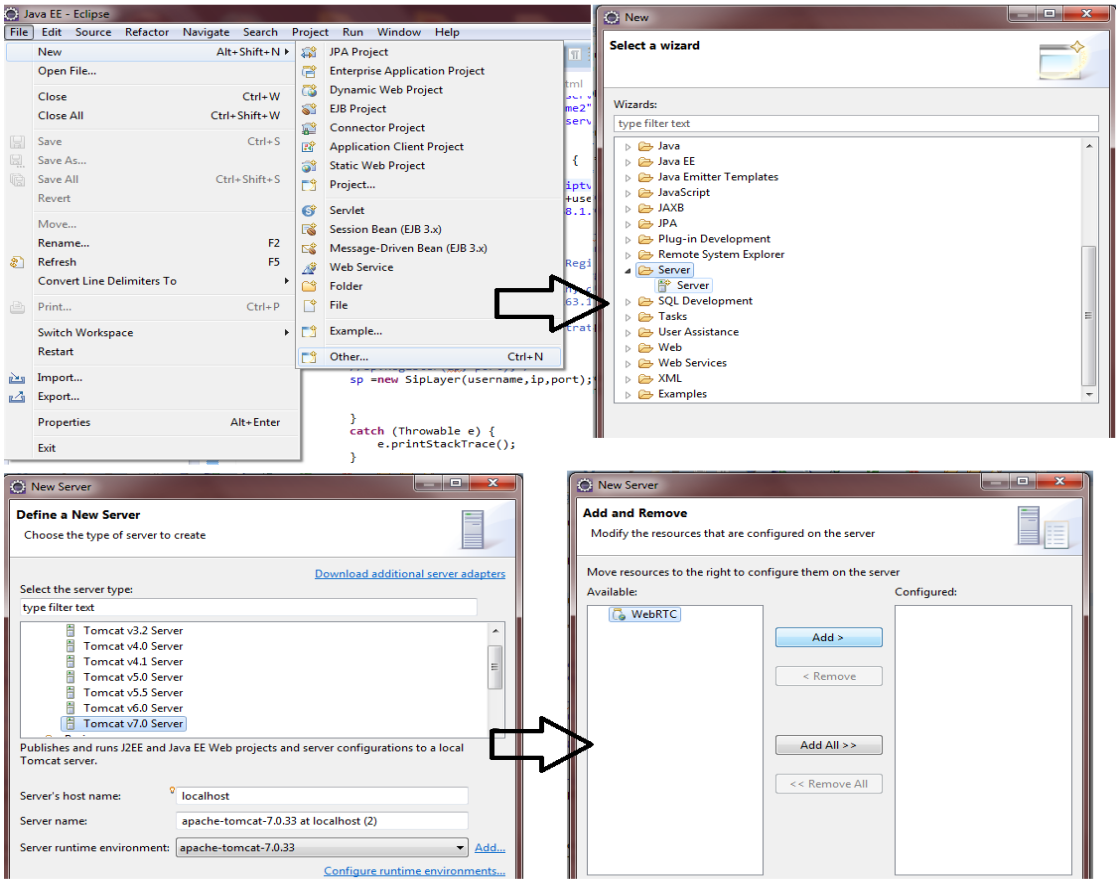

Ilustración 45 Cómo añadir Apache Tomcat a Eclipse

La compilación en eclipse es automática, por lo que una vez que hayamos realizado estos pasos, únicamente habrá que realizar un PLAY sobre el proyecto definido en Eclipse y desde un navegador acceder a la siguiente dirección:

http://localhost:8080/clienteWeb/interfazweb.html

Obteniendo la aplicación del cliente como se muestra en la Ilustración 29.

- Compilación y ejecución del Servidor SCF: como el servidor SCF está únicamente formada por clases Java (ver Tabla 4), podemos compilar y ejecutar directamente desde consola. Las carpetas que se muestran a continuación, han de estar contenidas en una carpeta denominada *ServidorSIP*:

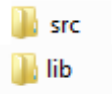

Ilustración 46 Carpetas en el servidor SCF

Dentro de la carpeta src ha de encontrarse otra, con el nombre rtc, donde se encuentren las clases Java. En lib almacenamos las librerías necesarias para la compilación y ejecución.

Para compilar:

C:/ServidorSIP> <ruta de la versión Java utilizada para compilar> javac lib/jainSipApi1.2.jar lib/jainSipRi1.2.jar lib/concurrent.jar lib/log4j-1.2.8.jar lib/commons-codec-20041127.091804.jar src/SipLayer.java src/Rtc.java

Para ejecutar:

C:/ServidorSIP> <ruta de la versión Java utilizada para compilar> javac lib/jainSipApi1.2.jar lib/jainSipRi1.2.jar lib/concurrent.jar lib/log4j-1.2.8.jar lib/commons-codec-20041127.091804.jar Rtc

Si preferimos, esta parte también puede ser compilada y ejecutada mediante Eclipse. En este caso realizando un *Java Proyect* en Eclipse y con la siguiente estructura de carpetas (incluimos las librerías JAIN SIP e IMS, igual que para el caso del cliente):

Ilustración 47 Proyecto servidor SCF en Eclipse

- Ejecución del servidor de vídeo: explicada con detalle en el apartado 0.

Anexo C: Configuración de VLC como servidor de vídeo

Para poder configurar la aplicación de VideoLAN como servidor VLC, los pasos a seguir en la configuración son los siguientes:

1. Abrimos la aplicación VLC Media Player

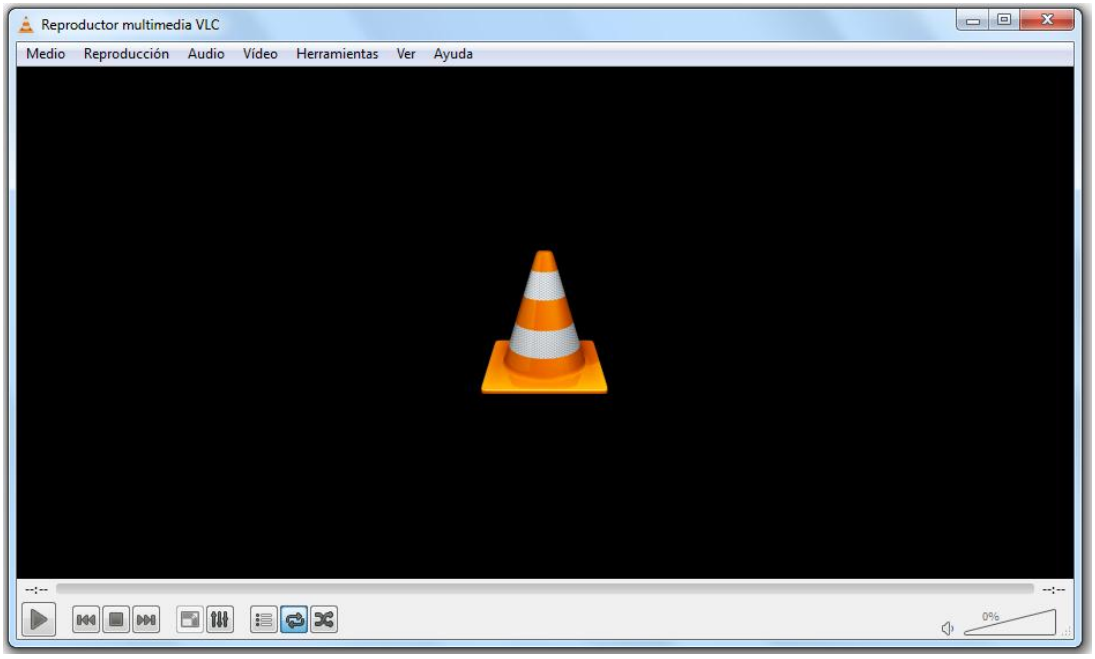

Ilustración 48 Aplicación VideoLAN

2. Pulsamos en el botón *Medio* de la barra de herramientas y en el menú desplegable pulsamos en *Emitir*

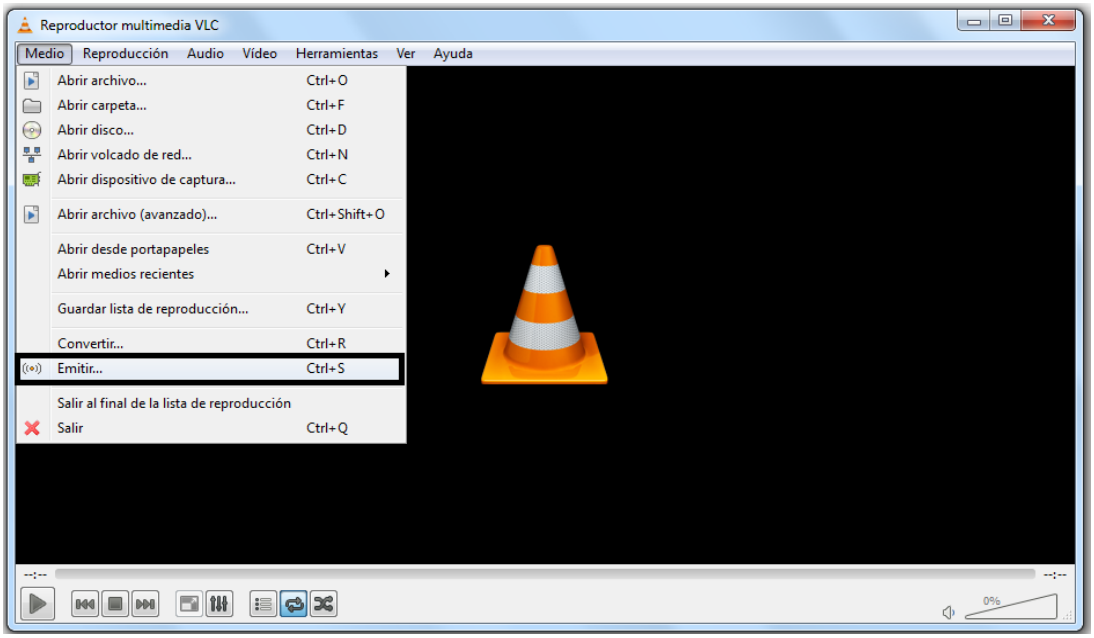

Ilustración 49 Abrir modo Emisión de archivo

3. Nos parece una nueva ventana, en la que pulsando el botón Añadir, indicaremos la ubicación del fichero de vídeo que queremos emitir. Tras añadirlo, pulsamos el botón *Emitir*.

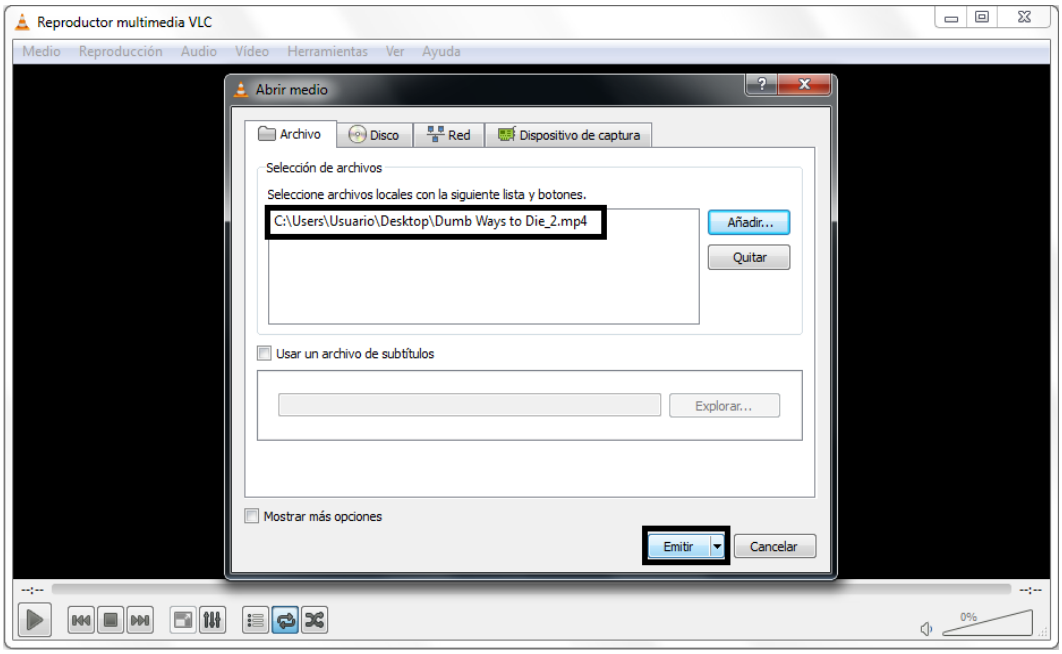

Ilustración 50 Selección de vídeo a emitir

4. Aparece una nueva ventana, denominada "*Salida de emisión*" en la que en uno de sus campos, denominado "*Fuente*" veremos la ubicación del fichero que será emitido como vídeo. Pulsamos en *Siguiente*.

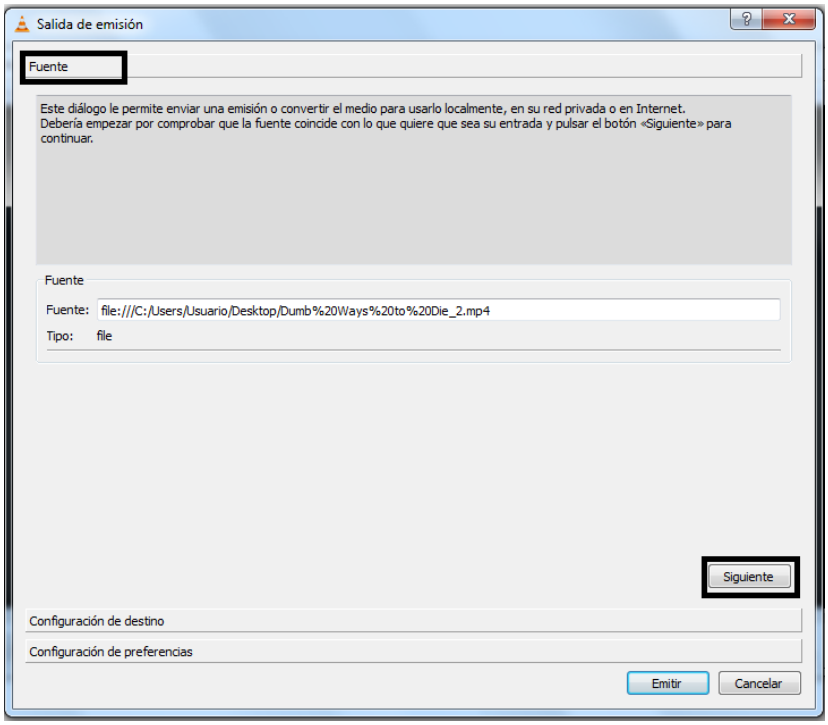

Ilustración 51 Fuente del vídeo

5. Pasamos a configurar ahora la "Configuración de destino" de nuestra emisión. Lo primero de todo es deshabilitar la transcodificación, ya que para este caso no es necesario. En "Archivo" desplegamos y seleccionamos RTSP y a continuación pulsamos el botón Añadir. Podemos también, como opción adicional, mostrar en local el archivo de vídeo que estamos emitiendo, sin más que seleccionar "Mostrar en local".

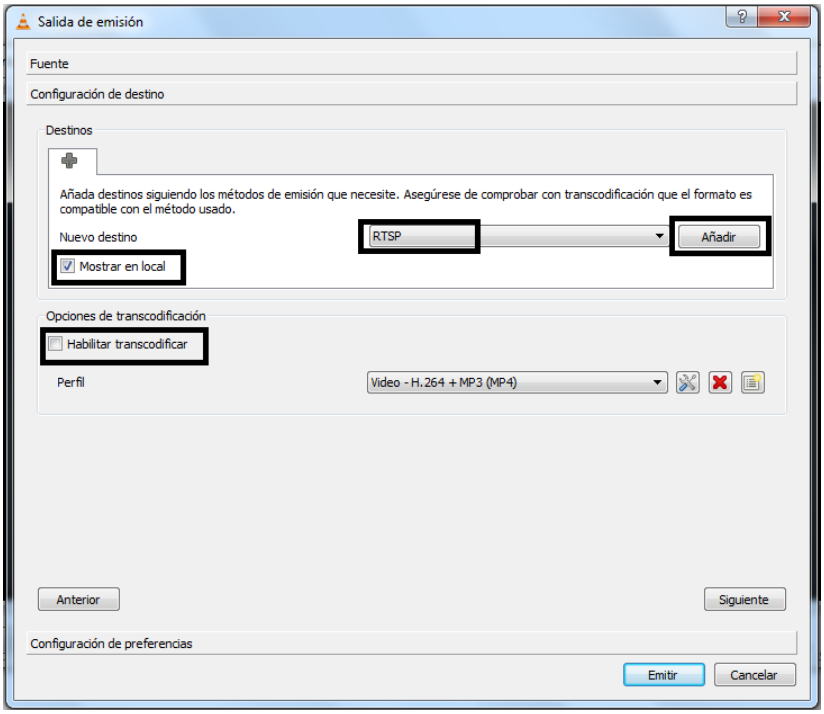

Ilustración 52 Definición del tipo de emisión, parte 1

6. Tras pulsar en *Añadir*, aparece una nueva ventana en la que configuraremos el puerto de emisión, en este caso el 5554, y la ruta, en la que pondremos *stream*. El nombre es indiferente, pero ha de coincidir con el nombre que se añade al final de la ruta del elemento embebido en la página HTML del cliente.

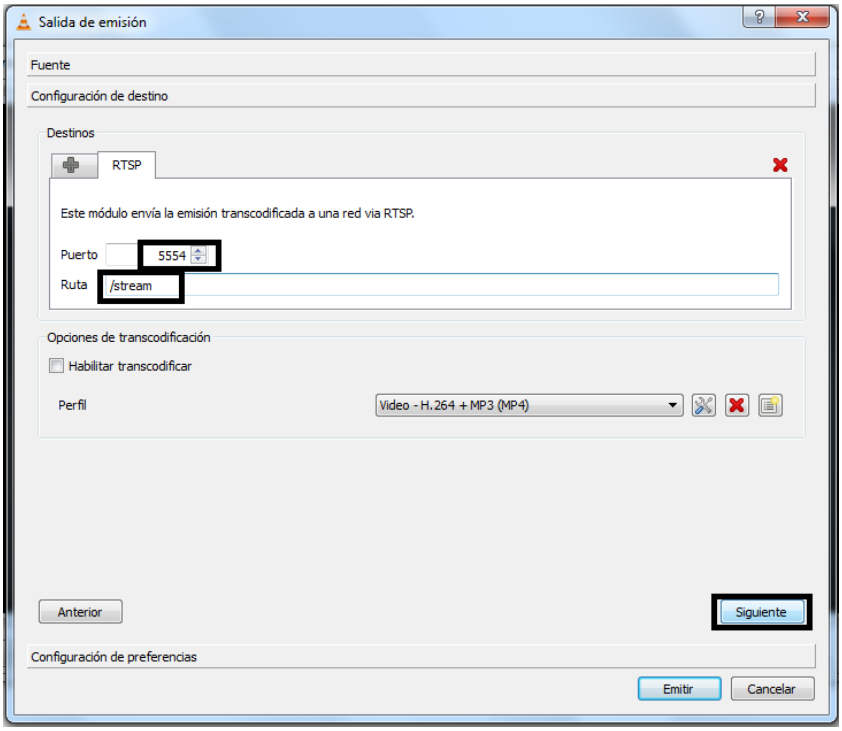

Ilustración 53 Definición del tipo de emisión, parte 2

7. Finalmente pulsamos en *Siguiente* y en *Emitir*.

Anexo D: Glosario

AS	*Application Server*
AJAX	*Asynchronous JavaScript And XML*
API	*Application Programming Interface*
ASF	*Apache Software Fundation*
BGCF	*Breakout Gateway Control Function*
B2BUA	*Back To Back User Agent*
CAMEL	*Customized Applications for Mobile network Enhaced Logic*
CoD	*Content on Demand*
COIT	Colegio Oficial de Ingenieros de Telecomunicación
C-SCF	*Call State Control Function*
ECMA	*European Computer Manufacturers Association*
ETSI	*European Telecommunications Standards Institute*
FTP	*File Transfer Protocol*
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile Communications*
HSS	*Home Subscriber Server*
HTML	*HyperText Markup Language*
HTTP/HTTPS	*Hypertext Transfer Protocol/Secure*
I-CSCF	*Interrogating-Call State Control Function*
IETF	*Internet Engineering Task Force*
iFC	*Initial Filter Criteria*
IGMP	*Internet Group Management Control*
IMS IP	*Multimedia Subsystem*
IP-MMG	*IP Multimedia Media Gateway*
ISO	*International Organization of Standardization*
JDK	*Java Development Kit*
JSEP	*Javascript Session Establishment Protocol*
JSPs	*Java Server Pages*
J2SE	*Java2ndStandardEdition*
MF	*Media Function*
MGCF	*Media Gateway Control Function*
MGW	*Media Gateway*
MMS	*Multimedia Messaging System*
MRFC	*Multimedia Resource Function Controller*
MRFP	*Multimedia Resource Function Processor*
NGN	*Next Generation Networks*
N-PVR	*Network-Personal Video Recorder*
OSA SCS	*Open Service Architecture Service Capability Server*
P-CSCF	*Proxy-Call State Control Function*
PSTN	*Public Switched Telephone Network*
PTT	*Push To Talk*
QoS	*Quality of Service*
RFC	*Request for Comments*
RTCP	*Real time Transport Control Protocol*
RTP	*Real Time Protocol*
RTSP	*Real Time Streaming Protocol*

SAP	*Session Announcement Protocol*
SCF	*Service Control Function*
S-CSCF	*Serving- Call State Control Function*
SDP	*Session Description Protocol*
SGW	*Signalling Gateway*
SIP	*Session Initiation Protocol*
SLF	*Subscription Location Function*
SMTP	*Simple Mail Transfer Protocol*
SS7	*Signalling System No.7*
TCP	*Transport Control Protocol*
TISPAN	*Telecommunications and Internet converged Services and Protocols for Advanced Networks*
UA	*User Agent*
UAC	*User Agent Clients*
UAS	*User Agent Servers*
UDP	*User Datagram Protocol*
UE	*User Equipment*
UMTS	*Universal Mobile Telecommunication System*
VCD	*Video Compact Disc*
VoIP	*Voice over IP*
WebRTC	*Web Real Time Communications*
W3C	*World Wide Web Consortium Protocols for Advanced Networking*
XMLHTTP	*Extensible Markup Language Hypertext Transfer Protocol*
3GPP/3GPP2	*Third Generation Partnership Project/2*

Parte VI: Bibliografía

Bibliografía

1. Cisco Visual Networking Index: Forecast and Methodology, 2012–2017. [En línea] http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html.

2. **TISPAN, ETSI.** *IPTV Functions Supported by the IMS Subsystem.* 2011.

3. **Tel09.** *Telecommunications and Internet Converged Services and Protocols for Advanced Networking (TISPAN): IMS-based IPTV stage 3 specification.* 2006-2009.

4. **J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, e. Schooler.** *SIP: Session Initiation Protocol RFC 3261.* June 2002.

5. The Internet Engineering Task Force. [En línea] http://www.ietf.org.

6. **M. Handley, V. Jacobson, C. Perkins,.** *SDP: Session Description Protocol.* Julio 2006.

7. **M. Handley, C. Perkins, E. Whelan.** *Session Announcement Protocol.* Octubre 2000.

8. **J. Rosenberg, H. Schulzrinne.** *An Offer/Answer Model with the Session Description Protocol (SDP).* Junio 2002.

9. **Sons, John Wiley &.** *The 3G IP Multimedia Subsystem (IMS).* 2008.

10. The 3rd Generation Partnership Project. [En línea] http://www.3gpp.org/.

11. **Subsystem, The IP Multimedia.** *Iván Vidal Fernandez, Ignacio Soto Campos.* s.l. : Master in Telematic Engineering slides, 2001-2012.

12. **V. Fajardo, J. Arkko, J. Loughney, G. Zorn.** *Diameter Base Protocol.* Octubre 2012.

13. **H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson,.** *RTP: A Transport Protocol for Real-Time Applications.* 2013.

14. **Prado, Fernando Cores.** *Arquitecturas Distribuidas para Sistemas de Vídeobajo-demanda a gran escala,Departamento de informática de la unidad de arquitectura de ordenadores y sistemas operativos, Universidad de Barcelona.* 2003.

15. **Tel11.** *Telecommunications and Internet converged Services and Potocols for Advanced Networking (TISPAN): IPTV functions supported by the IMS subsystem.* 2003-2011.

16. **H. Schulzrinne, A. Rao, R. Lanphier.** *Real Time Streaming Protocol (RTSP).* 1998.

17. WebRTC. [En línea] http://www.webrtc.org/.

18. **Alan B. Johnston, Daniel C. Burnett.** *WebRTC: APIs and RTCWeb protocols of the HTML5 Real-Time Web.*

19. JAIN SIP Specification. [En línea] http://jcp.org/en/jsr/detail?id=032.

20. **Phelim O´Doherty, Mudumbai Ranganathan.** JAIN SIP Tutorial. [En línea] http://yassine.ab.free.fr/Documents/JAIN-SIP-Tutorial.pdf.

21. **Apache.** Apache Tomcat. [En línea] http://tomcat.apache.org/index.html.

22. **w3schools.** Tutorial Javascript. [En línea] http://www.w3schools.com/js/.

23. **w3schoolsH.** HTML5 Introduction. [En línea] http://www.w3schools.com/html/html5_intro.asp.

24. **3schoolsX.** XML Tutorial. [En línea] http://www.w3schools.com/xml/.

25. **Java.** ¿Qué es Java? [En línea] http://www.java.com/es/.

26. Interface Servlet. [En línea] http://tomcat.apache.org/tomcat-7.0-doc/servletapi/javax/servlet/Servlet.html.

27. w3schoolsAjax. [En línea] http://www.w3schools.com/ajax/ajax_intro.asp.

28. **Gómez, Francisco Javier de Pablos.** *Desarrollo de un servicio P2P TV para redes de Internet de próxima generación.* s.l. : ,Proyecto Fin de Carrera Universidad Carlos III de Madrid, Marzo 2012.

29. Open IMS Core. [En línea] [Citado el: 21 de Octubre de 2013.] http://www.openimscore.org.

30. VideoLAN. [En línea] http://www.videolan.org/vlc/.

31. **COIT.** *El ingeniero de Telecomunicación: perfil Socio- Profesional.* Madrid : s.n., Febrero 2013.

32. **A. Johnston, S. Donovan, R. Sparks, C. Cunningham, K. Summers.** *Session Initiation Protocol, Basic Call Flow Examples.* December 2003.

33. European Telecommunications Standards Institute. [En línea] http://www.etsi.org/.

34. VideoLANs Wiki. [En línea] http://wiki.videolan.org.

35. APIs WebRTC. [En línea] http://www.webrtc.org/reference/native-apis.

36. **WebRTC, HTML5.** Getting Started with WebRTC. [En línea] http://www.html5rocks.com/en/tutorials/webrtc/basics/.

37. **StrEduc.** ¿Qué es el Streaming? [En línea] http://www.ite.educacion.es/formacion/materiales/107/cd/video/video0103.html.

38. MarketingDirecto. [En línea] http://www.marketingdirecto.com/especiales/marketing-movil/espana-lider-europea-en-el-uso-de-smartphones-con-un-552-de-penetracion/.

39. MarketingDirecto http://www.marketingdirecto.com/especiales/marketing-movil/espana-lider-europea-en-el-uso-de-smartphones-con-un-552-de-penetracion/. [En línea] http://www.marketingdirecto.com/especiales/marketing-movil/espana-lider-europea-en-el-uso-de-smartphones-con-un-552-de-penetracion/.